TRABAJO DE FIN DE GRADO

CURSO 2013-2014

Algoritmo de Identificación de Zonas Geográficas Basado en el Contenido de la Imagen

Jesús Martín Alonso

Directores:

Luis Javier García Villalba

Ana Lucila Sandoval Orozco

FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

El abajo firmante autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Grado: "Algoritmo de Identificación de Zonas Geográficas Basado en el Contenido de la Imagen", realizado durante el curso académico 2013-2014 bajo la dirección de Luis Javier García Villalba y Ana Lucila Sandoval Orozco en el Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

_____

Jesús Martín Alonso

Agradecimientos

A mis directores Luis Javier García Villalba y Ana Lucila Sandoval Orozco por haberme dado la oportunidad de trabajar con ellos, por. haber hecho posible la realización de este trabajo fin de grado y por toda su dedicación durante estos meses de duro trabajo.

A todos mis compañeros de la Facultad por vuestra amistad y vuestros ánimos.

Mi más especial agradecimiento está dedicado a mi familia por su apoyo, cariño y esfuerzo que me han permitido realizar todos mis estudios.

Abstract

The present work pretends to be an introduction to the forensic analysis of digital images of mobile devices. The work focuses on the identification of an area making use of the content of the image. Firstly, a revision of the main methods of detection and description of interest points in an image is made. Then, an algorithm for identifying a geographic area using SIFT descriptors for features and keypoints extraction of the image is presented. In the process of identification, we create a database of images of the area under investigation with downloaded images from Google Street View to perform, then, the search and identification of the area. Finally, several experiments in order to evaluate the performance of the algorithm were performed.

Keywords.

Digital image, forensics analysis, Python, SIFT descriptors

Resumen

El presente trabajo pretende ser una introducción al análisis forense de imágenes digitales de dispositivos móviles. El trabajo se centra en la identificación de una zona a partir del contenido de la imagen. Primero se realiza una revisión de los principales métodos de detección y descripción de puntos de interés en una imagen. A continuación, se presenta un algoritmo de identificación de una zona geográfica que utiliza los descriptores SIFT para la extracción de características y puntos clave de la imagen. En el proceso de identificación se crea una base de datos de imágenes de la zona objeto de investigación con imágenes descargadas desde Google Street View para realizar, a continuación, la búsqueda e identificación de la zona. Finalmente, se realizó una serie de experimentos para evaluar el funcionamiento del algoritmo.

Palabras clave

Análisis forense, descriptores SIFT, imagen digital, Python.

Lista de Acrónimos

API	Application Programming Interface
EXIF	Exchangeable Image File Format
FAST	Features from Accelerated Segment Test
FLANN	Fast Library for Approximate Nearest Neighbors
GLOH	Gradient Location and Orientation Histogram
GPS	Global Positioning System
HOG	Hessian Of Gaussian
HTML	HyperText Markup Language
RANSAC	RANdom SAmple Consensus
SFM	Structure For Motion
SIFT	Scale Invariant Features Points
SURF	Speeded Up Robust Features
SUSAN	Smallest Univalue Segment Assimilating Nucleus
XML	eXtensible Markup Language

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

1.1. Motivación

Hoy en día es común el uso de teléfonos móviles inteligentes con el que se pueden tomar fotos de alta calidad y almacenar información relevante sobre la captura de la imagen en metadatos Exif. La inclusión de información para geo-localización en metadatos Exif, convierte en un proceso trivial la identificación del punto exacto desde el cual fue tomada una fotografía. Sin embargo, existen escenarios en los que es necesario contar herramientas que ayuden a realizar un análisis de imágenes que no contienen metadatos con el objetivo de identificar el lugar donde fue tomada una foto:

- La mayoría de usuarios de teléfonos móviles desactivan los dispositivos GPS haciendo imposible el almacenamiento de información de geolocalización en los metadatos Exif.

- El creciente uso de las redes sociales como Facebook, Twitter o Flickr, en las que se pueden publicar fotos directamente de los dispositivos móviles hace necesaria la optimización del almacenamiento requerido por cada imagen. Entre otros procesos, eliminando los metadatos Exif cuando un usuario sube una imagen.

Por tanto, es necesario utilizar el contenido de la imagen como herramienta para realizar la tarea de geo-localización. El procesamiento digital de imágenes permite analizar y manipular imágenes en un ordenador para obtener información de una determinada escena captada por una cámara fotográfica digital. Esta información permite, entre otras aplicaciones, mejorar la calidad, restaurar una imagen, detectar alteraciones en la imagen, medir características geométricas y de color de diferentes objetos, detección de la presencia de características, realizar un seguimiento de patrones. Sin embargo, la extracción

de información de interés en una imagen digital es uno de los principales retos del campo del procesamiento de imagen.

Como resultado de este análisis la motivación principal de este proyecto es el diseño e implementación de una herramienta de apoyo que permita identificar el lugar donde fue tomada una fotografía utilizando herramientas de procesamiento digital de imágenes para detectar características que permitan su localización y delimitando el área de análisis a una región urbana específica.

1.2. Objetivos

El presente Trabajo de Fin de Grado (TFG) tiene los siguientes objetivos:

- Realizar un estudio de las técnicas de extracción de características de imágenes digitales con objeto de clasificar las propuestas más relevantes.
- Presentar las principales técnicas de análisis forense que determinan la geolocalización de una imagen digital.
- Diseñar e implementar en el lenguaje de programación Python un algoritmo que permita obtener la posición geográfica imagen digital utilizando técnicas de extracción de características.

1.3. Estructura

La memoria está organizada en 6 capítulos siendo el primero la presente introducción.

En el capítulo 2 se hace una introducción de las principales técnicas de detección y extracción de características de imágenes y del servicio de Street View.

El capítulo 3 presenta un estudio del uso de Street View y descriptores de

imágenes en la localización de imágenes.

El capítulo 4 presenta una descripción del método desarrollado para la geolocalización de imágenes basado en el uso de descriptores Sift y Street View. En el capítulo se detallan las fases de funcionamiento del mismo.

El capítulo 5 presenta los experimentos realizados para evaluar el funcionamiento del algoritmo.

En el capítulo 6 se presenta las principales conclusiones de este trabajo.

2. TÉCNICAS DE DETECCIÓN Y DESCRIPCIÓN DE IMÁGENES.

En este capítulo se presenta el uso de los puntos claves o características y descriptores dentro del contexto de procesamiento de imágenes digitales y el uso del servicio de Street View como herramienta para la obtención de dichas imágenes. Las secciones 2.1 a 2.5 se presentan una introducción a los conceptos de puntos claves o características y descriptores explicando su uso y una clasificación de los mismos. En la sección 2.6 se presenta una descripción del servicio de Street View de Google Maps.

2.1. Características de una Imagen

En procesamiento de imágenes digitales los puntos clave son puntos donde se encuentra una característica, por lo que a menudo, ambos conceptos se usan indistintamente. De esta manera, las características o puntos clave de una imagen se definen como patrones que pueden ser fácilmente encontrados, únicos, y fácilmente comparables. A pesar que no hay una definición exacta de qué constituye una característica, se puede definir como una parte interesante de una imagen. Los descriptores, por su parte, contienen información en una región alrededor del punto clave, de tal manera que permitan comparar las características de dos imágenes.

Las características de una imagen tienen gran cantidad de aplicaciones en la vida real. Entre otras, se encuentran las siguientes:

- Reconocimiento y categorización de objetos [1][2].
- Reconstrucción tridimensional (3D) [3].
- Detección de movimiento[4].
- Reconocimiento de modelos[5]
- Generación de panorámicas [6]

Para explicar el uso de las características en la generación una imagen panorámica se presenta el siguiente ejemplo:

Partiendo de varias fotos tomadas de un paisaje se desea generar una foto panorámica como se observa en la Figura 2.1.

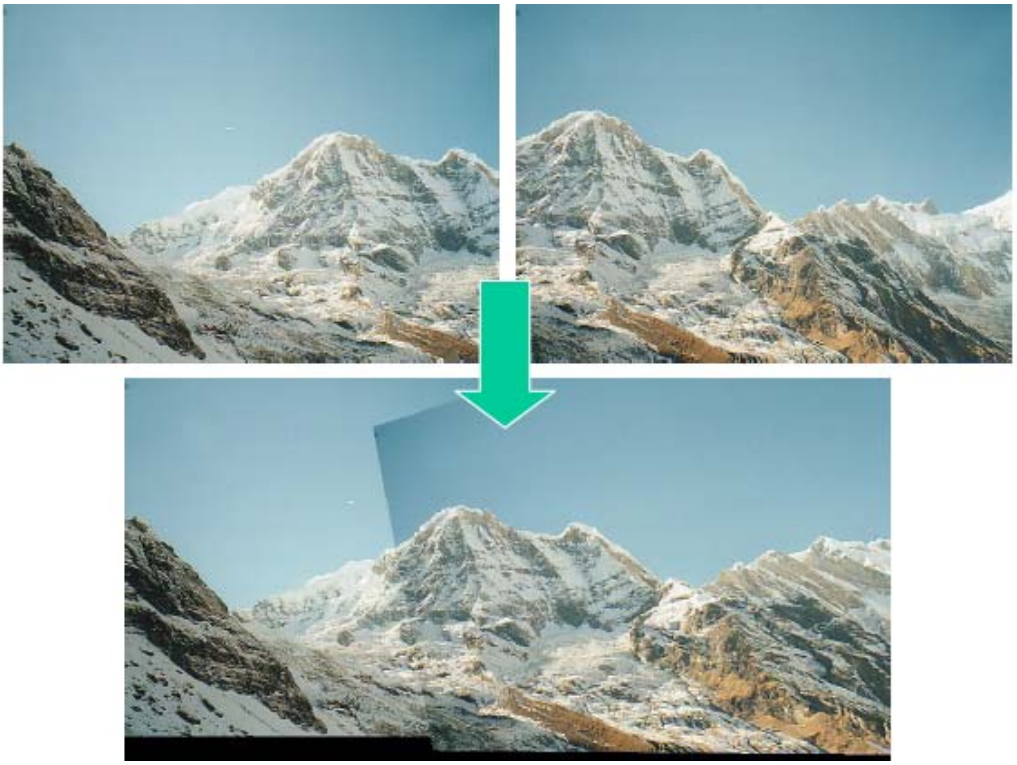


Fig. 2.1. Obtención de una panorámica a partir de 2 imágenes contiguas.

El primer paso es detectar las características en ambas imágenes y comparar las regiones de dichas imágenes usando dichas características. La Figura 2.2 ilustra este proceso.

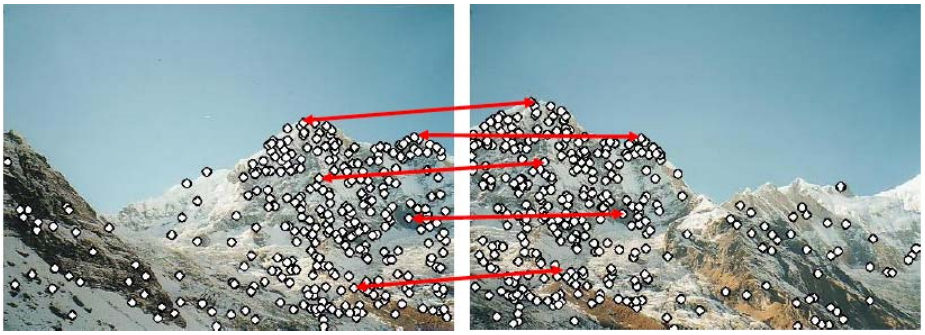


Fig. 2.2. Comparación de imágenes usando características.

2.2. Detección de Características

La búsqueda de características o puntos clave en una imagen digital se puede ver como la resolución de un puzle, en los que se debe formar una imagen, a partir de cientos de piezas. Sin embargo, nunca se pregunta el proceso que se sigue para resolver el puzle. Si se observa detenidamente, se realiza un proceso de búsqueda de características, patrones, … que sean fáciles de encontrar en las piezas y fácil de comparar. Dada la complejidad en la definición de dichas características, se utilizará el siguiente ejemplo para definirlas.

En la Figura 2.3 se muestran 6 piezas (A-F) pertenecientes a la imagen del edificio.

Fig. 2.3. Ejemplo de características en una imagen

A y B son superficies planas (A es parte del cielo y B es parte de la pared del edificio) por lo que es muy difícil saber cuál es la posición exacta de estas piezas en la imagen original. C y D son bordes del edificio. Se puede saber aproximadamente de donde es la pieza, pero al repetirse la pieza, no se podrá definir su posición exacta. Por último, E y F son esquinas que inmediatamente se pueden localizar con exactitud, ya que son únicas.

En la Figura 2.4 se puede ver lo mismo pero de manera más simple. Así A y B serían el cuadrado azul, C y D el negro y E y F el rojo. Así se observa que si se mueve el cuadrado azul, no cambia. Si se mueve el cuadrado negro horizontalmente no cambia, pero sí cambia al moverlo verticalmente. Por último se ve que el cuadrado rojo cambia tanto si se mueve vertical como horizontalmente.

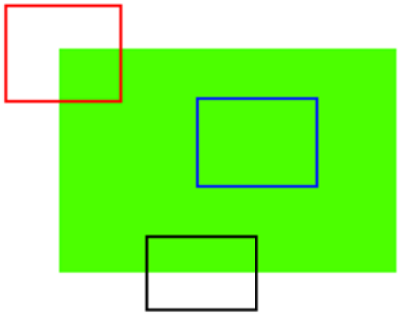


Fig. 2.4. Esquema de las posibles características

De esta manera, se puede definir que una característica es un patrón que puede ser fácilmente encontrado, único, y fácilmente comparable.

Así, se pueden encontrar puntos claves óptimos buscando las regiones que tienen la máxima variación cuando se mueven horizontal y verticalmente. La detección de estos puntos clave recibe el nombre de **"Detección de Características"** [7].

La Figura 2.5. ilustra los tres casos descritos anteriormente. La ventana de la imagen de la izquierda no cambia al moverla, mientras que la ventana de la imagen central sólo cambia al moverla horizontalmente y la ventana de la derecha cambia al realizar cualquier movimiento.

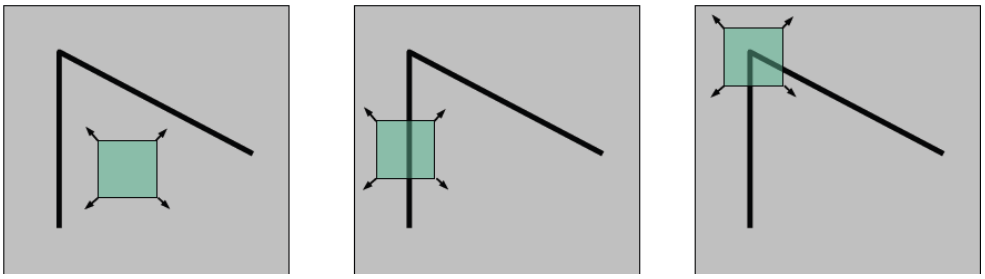


Fig. 2.5 Casos posibles en la detección de características

Una vez encontradas las características, se pretenderá encontrarlas en otras imágenes, en nuestro caso, para ver si las imágenes tienen algo en común. Para ello, se actual de la misma manera que actuaría una persona: Describiendo la región alrededor del punto clave, explicándolo como por ejemplo, para el parche D: "La parte de arriba es cielo azul, la parte de abajo son cristales del edificio, …". Esta descripción recibe el nombre de "**Descripción de Características**"[7] [8].

2.3. Tipos de Características

Como se ha comentado anteriormente, definir lo que es una característica es complicado, pudiendo definirse como una parte o región de la imagen que resulta "interesante" por sus características. Se pueden clasificar los tipos de características en las siguientes categorías:

- **Bordes (Edges):** Se pueden definir los bordes como puntos que separan dos regiones en una imagen. También se pueden definir bordes como puntos en los que el brillo de la imagen cambia de manera notoria.

- **Esquinas (Corners):** Como se ha visto anteriormente, las esquinas son buenas características, ya que ofrecen un alta variación al mover la región. Formalmente las esquinas pueden ser definidas como la intersección de dos bordes. A menudo, se habla de "esquinas" como "puntos de interés".

- **Regiones (Blob):** Las regiones son áreas o partes de la imagen. De esta

manera los detectores y descriptores de este tipo de características describen la imagen en función de regiones, al contrario que los que detectan características anteriores que describen en función de puntos.

De esta manera se pueden clasificar los detectores de características más conocidos según el tipo de características que detectan [9].

2.4. Detectores de Características

A continuación se exponen las distintas técnicas de detección de características, enumerando los detectores más utilizados de cada caso.

2.4.1. Detección de Bordes

La detección de bordes es el conjunto de métodos matemáticos que tiene como objetivo encontrar bordes, es decir, puntos en los que el brillo de la imagen cambia drásticamente.

La detección de bordes es una herramienta fundamental en procesamiento de imágenes y visión computacional. Está demostrado que los cambios en el brillo de la imagen (bordes) se corresponden a menudo con:

- Cambios de profundidad.
- Cambios en la orientación de la superficie.
- Cambios en las propiedades del material
- Variaciones en la iluminación de la escena.

De esta manera, la aplicación de un detector de bordes a una imagen puede revelar información interesante (límites de objetos, …) que puede servir para reducir la cantidad de datos a procesar [10] (véase Figura 2.6.)

Fig. 2.6. Resultado de aplicar un detector de bordes (Canny) a una imagen.

Detectores de bordes conocidos:

- Canny
- Canny-Deriche
- Differential
- Sobel
- Prewitt
- Roberts Cross

2.4.2. Detección de Esquinas

La detección de esquinas hace referencia al conjunto de métodos matemáticos y algoritmos para identificar las esquinas de una imagen. La detección de esquinas tiene gran cantidad de aplicaciones en el campo de la visión computacional, como el modelado 3D, la detección de movimiento o el reconocimiento de objetos [11].

Detectores de esquinas más conocidos:

- Harris & Stephens
- Shi-Tomasi
- SUSAN
- FAST

2.4.3. Detección de Regiones

La detección de regiones hace referencia a los métodos matemáticos que tienen como objetivo la detección de regiones en una imagen digital que difieren en propiedades, como brillo o color, comparadas con áreas que rodean a dichas regiones. Informalmente, una región es un área de la imagen en la que algunas propiedades son constantes o varían dentro de un rango determinado de valores. Por ejemplo, todos los puntos en una burbuja pueden ser considerados de ser similares entre sí [12].

La motivación de este tipo de detectores es que proporcionan información que no pueden proporcionar los detectores de bordes y esquinas.

Un problema de los detectores de esquinas vistos anteriormente es que no son invariantes con respecto a la escala. Esto significa que las características, en este caso esquinas, de una misma escena con distintos niveles de zoom (escala) no serán detectadas.

En la Figura 2.7. se ve la misma esquina representada en dos escalas distintas. Sin embargo, al pasarle el detector, en la imagen de la izquierda clasificará las características como bordes y no como esquinas, mientras que en la imagen de la derecha si la clasificará como esquina. Los detectores de regiones que se exponen a continuación, solucionan este problema.

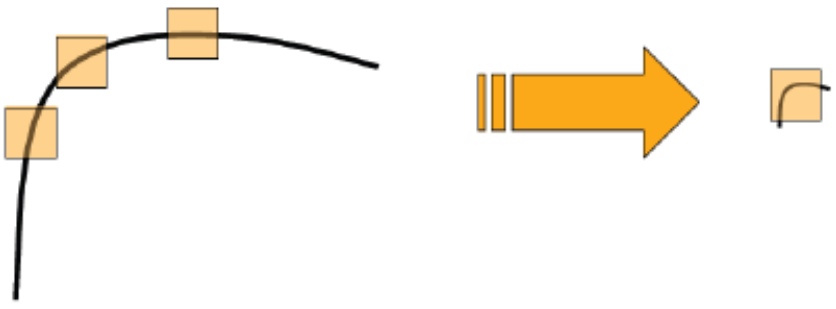


Fig. 2.7. Un borde puede convertirse en esquina al cambiar la escala.

Se distinguen dos clases principales de detectores de regiones: *métodos diferenciales,* basados en derivadas de la función con respecto a la posición y *métodos extremos*, que están basados en encontrar máximos y mínimos locales de la función.

Los detectores de regiones más comunes y utilizados son los siguientes:

- La Laplaciana de Gauss
- La Diferencia de Gauss
- El Determinante de Hessian
- Hessian-Laplace

Seguidamente se describen en detalle cada uno de ellos:

La Laplaciana de Gauss

Uno de los detectores de regiones más utilizados es el llamado operador Laplaciana de Gauss. La ventaja de este detector es que sus características detectadas son invariantes por cambios de escala. Para conseguir esta propiedad este detector incorpora la función de Gauss bidimensional, para introducir el parámetro σ que representa la escala [12].

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma} e^{-(x^2+y^2)/(2\sigma)}$$

Así, se construye la función *espacio-escala* L, que se define como la convolución de *G*, con *f(x,y)*, siendo esta última la función que define a la imagen [11].

$$L(x, y, \sigma) = G(x, y, \sigma) * f(x, y)$$

De esta manera, se tiene una representación "escala-espacio", es decir, para diferentes valores de σ tenemos la imagen en diferentes escalas. Esta función es la base de todos los operadores detectores que tienen la propiedad de ser

invariantes por los cambios de escala.

En concreto este detector utiliza la Laplaciana de la función *escala-espacio* que viene dada por la suma de las derivadas segundas de *L* [12].

$$\nabla^2 L = L_{xx} + L_{yy}$$

Así, para detectar los puntos de interés con su escala correspondiente, se encontrarán los máximos y mínimos de la laplaciana. El problema es que dependen de la relación entre el tamaño de la imagen y la escala, para evitar este problema se utiliza la laplaciana normalizada [12]:

$$\nabla^2_{norm} L(x, y, \sigma) = \sigma(L_{xx} + L_{yy})$$

Y detectar los máximos y mínimos de esta función

$$(x_0, y_0, \sigma_0) = \arg\max \min_{(x,y,\sigma)} (\nabla^2_{norm} L(x, y, \sigma))$$

que proporcionarán las regiones de interés.

La diferencia de Gauss [12]

La diferencia de Gauss es una aproximación a la laplaciana de Gauss, por lo que obtiene resultados muy similares. Se obtiene como el límite de la diferencia entre dos imágenes suavizadas Gaussianamente.

$$\nabla^2 L(x, y, \sigma) \approx \frac{L(x, y, k\sigma) - L(x, y, \sigma)}{k\sigma - \sigma}$$

Así se define la diferencia gaussiana como:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

A partir de esta aproximación se trabaja obteniéndolos extremos locales de la

función $D(x, y, \sigma)$. Este detector es el usado por el descriptor SIFT, por lo que se explica en profundidad más adelante.

El determinante del Hessiano.

La matriz Hessiana de la función escala-espacio L también se pude utilizar para detectar puntos característicos. En concreto, el determinante de la matriz Hessiana de *L* [12]:

$$\det HL(x, y, \sigma) = \sigma^2 (L_{xx} L_{yy} - L_{xy}^2)$$

$$(x_0, y_0, \sigma_0) = \arg\max \min_{(x,y,\sigma)} (\det HL(x, y, \sigma))$$

Permite encontrar los puntos clave calculando los máximos locales de dicho determinante.

El híbrido Hessiano - Laplaciano.

Esta función es un operador híbrido entre el Laplaciano y el determinante del Hessiano. La idea es encontrar los puntos clave en el espacio mediante el determinante del Hessiano y, para estos puntos seleccionados, se encuentra la escala maximizando en la escala σ mediante el Laplaciano [13]:

$$(x_0, y_0) = \arg\max \min_{(x,y)} (\det HL(x, y, \sigma))$$

$$\sigma_0 = \arg\max \min_{\sigma} (\nabla^2 L(x_0, y_0, \sigma))$$

Este operador ha sido usado para detección de imágenes, reconocimiento de objetos y análisis de texturas.

La Tabla 2.1 presenta un resumen de los principales detectores de características.

Detector de Características	Bordes	Esquinas	Regiones
Canny [14]	X		
Harris & Stephens / Plessey[15]	X	X	
SUSAN	X	X	
Shi & Tomasi		X	
FAST		X	X
Laplaciana de Gauss		X	X
Diferencia de la Gausiana		X	X
Determinante del Hessiano		X	X

Tabla 2.1. Tipos de detectores de características.

2.5. Descriptores de Características

Como se ha comentado la detección de características tiene gran cantidad de aplicaciones. Sin embargo, en muchos casos no solo se necesita detectar las características en una imagen, sino compáralas con las encontradas en otra imagen. Es el caso de nuestro propósito, en el que se tiene que comparar la imagen que queremos geolocalizar, con las imágenes que tenemos en la base de datos. A los algoritmos que se encargan de esta tarea se les llaman "descriptores de características" (Feature descriptors).

Existen gran cantidad de posibilidades para ello. La opción más simple es coger un área cuadrada de píxeles alrededor de cada característica o punto de interés detectado.

Se supone que la imagen tomada que queremos geolocalizar está en la base de datos. Dichas imágenes representan lo mismo pero pueden diferir en diferentes parámetros, como la orientación o el zoom. Así se necesita que tanto los detectores de características como los descriptores de características sean invariantes con respecto a estos parámetros.

El detector Harris, que detecta esquinas, es invariante respecto a cambios en la posición y orientación, puesto que las esquinas siguen siendo esquinas

aunque las giremos. Sin embargo, este detector no es invariante con respecto a los cambios en la escala.

Los descriptores de características recogen gran cantidad de información alrededor del punto de interés, por lo que se tiene que conseguir que sean invariantes con los distintos parámetros [8].

Los descriptores de características más conocidos y usados son los siguientes:

- SIFT (Scale Invariant Feature Transform)
- SURF
- GLOH
- HOG

2.5.1. Descriptores SIFT

El objetivo de los descriptores SIFT [16] es obtener características invariantes de imágenes que puedan ser usadas para la comparación fiable de distintas vistas de un objeto u escena.

Para ello las características deben ser altamente distintivas, lo que permite encontrar la correspondencia de una determinada característica en una base de datos con un gran número de características.

El algoritmo propuesto sigue las siguientes fases o etapas:

1. **Detección de los extremos escala-espacio.** Se encuentran puntos de interés o características que son invariantes con respecto a la escala y la orientación usando la diferencia de Gauss (DoG).

   El descriptor SIFT usa una aproximación de la *Laplaciana de Gauss,* la *Diferencia de Gauss,* por ser menos costosa computacionalmente hablando.

   El proceso de la búsqueda de extremos locales se lleva a cabo de la siguiente manera:

Para un primer valor de escala sigma (primera octava) se calculan la función escala-espacio mediante la convolución de la imagen con la función gaussiana con sigma en 5 niveles.

En cada escala se calcula la diferencia de Gauss entre dos niveles consecutivos (véase Figura 2.8.).

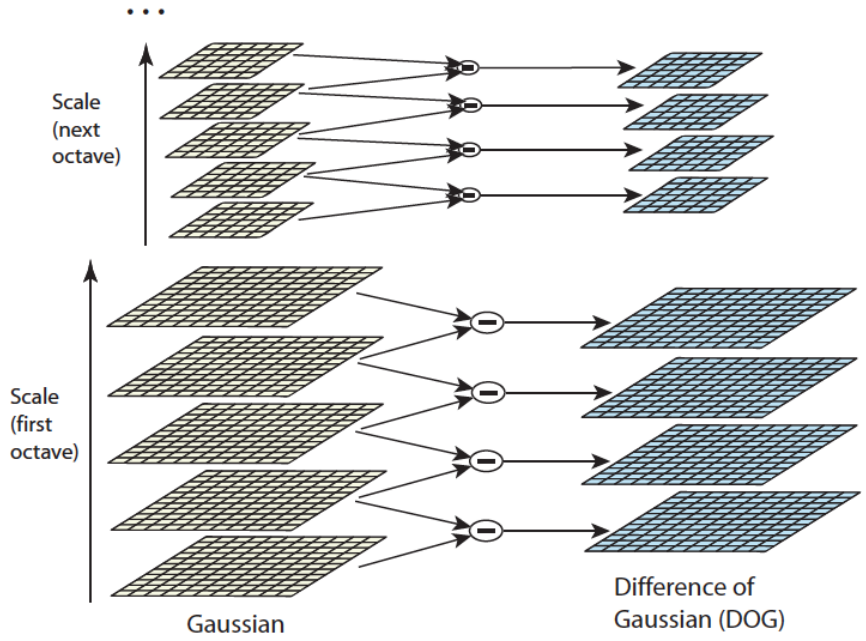


Fig. 2.8. Obtención de las Diferencias de Gauss

Una vez que se obtienen las *Diferencias de Gauss,* se buscan los extremos locales sobre la escala y el espacio. Por ejemplo, un pixel en una imagen es comparado con sus 8 vecinos y también con los 9 píxeles de la siguiente escala y los 9 píxeles de la escala anterior. Si es un extremo local, significa que el punto clave es mejor representado en esa escala (véase Figura 2.9.).

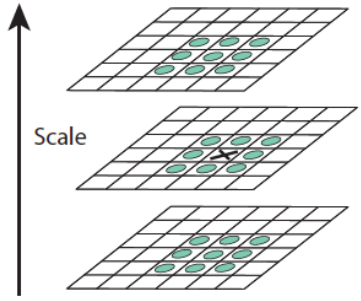


Fig. 2.9. Comparación de un píxel con sus vecinos

Cada matriz representa píxeles en una determinada escala. La X de la matriz del centro representa al potencial punto clave, que es comparado con todos los puntos verdes (píxeles vecinos).

2. **Extracción de los puntos clave**. Para cada característica candidata se determina la localización y la escala. Una vez que los potenciales puntos clave han sido localizados, se debe hacer un filtrado con el objetivo de obtener mejores resultados. A continuación, se seleccionan los puntos clave en base a su estabilidad.

3. **Asignación de la orientación de los puntos claves**: Para obtener invariancia con respecto a la rotación de la imagen, se asigna una orientación a cada punto clave detectado. Esta orientación mide hacia donde se produce más variación.

4. **Creación de descriptores**: Se crea el descriptor de cada punto clave. Para ello, se coge una matriz cuadrada de 16x16 alrededor de cada punto clave. A continuación, cada bloque se divide en un subbloque de 4x4. Para cada uno de ellos se crea un histograma de orientación. El proceso se ilustra en la Figura 2.10.

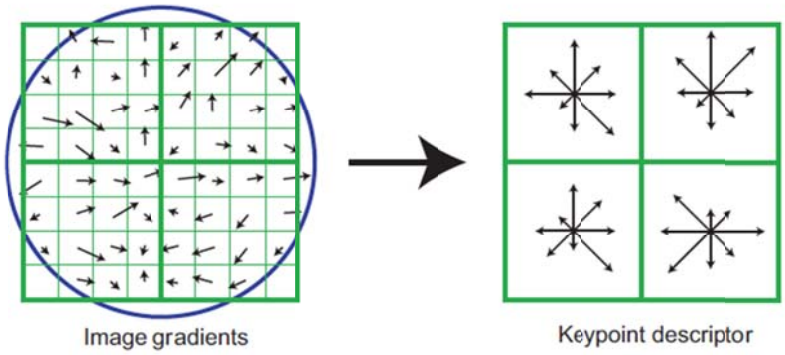


Fig. 2.10. Formación de un descriptor de un punto clave

Además, se guardan varias medidas más para conseguir mejores resultados frente cambios de luz, rotación.

2.5.2. Descriptores SURF

Los descriptores SURF [17] son diseñados con el objetivo de ser más rápidos que SIFT, sin perder fiabilidad. De hecho, están basados en SIFT.

Para conseguir esa mejora en el rendimiento introduce ciertos cambios:

- Para la detección de características utiliza el Determinante del Hessiano en vez de la Diferencia de Gauss usada por los descriptores SIFT. El Determinante del Hessiano se calcula de manera muy rápida utilizando solo 3 operaciones enteras.

- Para la asignación de orientación a cada punto clave y la construcción del descriptor se utiliza la suma de la respuesta del wavelet de Haar [18] alrededor del punto clave. Esta suma se calcula de manera eficiente y rápida con la ayuda de la imagen integral [19], algoritmo para realizar sumas sobre cuadrados.

2.5.3. Descriptores GLOH

Los descriptores GLOH al igual que SURF, están basados en SIFT. De hecho los primeros pasos del algoritmo son idénticos, desde la detección de los puntos de

interés hasta la asignación de orientación a cada punto clave.

La diferencia reside en la creación de los descriptores de cada punto clave. En este paso crea por cada punto clave crea un vector de 272 dimensiones que posteriormente se reduce mediante Análisis de Componentes Principales [20].

2.5.4. Descriptores HOG

Al contrario que los SURF y GLOH, los descriptores HOG [21] no están basados en SIFT. El principal objetivo de los descriptores HOG es el reconocimiento de objetos, obteniendo muy buenos resultados con el reconocimiento de personas.

La implementación se basa en dividir la imagen en pequeñas regiones conectadas, llamadas celdas, y para cada una de ellas obtener un histograma de las direcciones del gradiente para los píxeles de la celda. La combinación de estos histogramas representa el descriptor.

Los descriptores HOG son invariantes a transformaciones geométricas y fotométricas, pero no son invariantes con respecto a la orientación.

2.6. Google Street View

Una vez presentados los diferentes detectores y descriptores de características de imágenes digitales. Es indispensable conseguir una base de datos urbana para geo-localizar las imágenes. En internet se pueden encontrar grandes cantidades en redes sociales como Flickr, Facebook, entre otras. Flickr, por ejemplo, permite a los usuarios subir fotos y compartirlas, ofrecen millones de fotografías de todo el mundo pero es una base de datos poco uniforme, ya que la mayoría de las fotografías urbanas son de zonas turísticas. Por tanto, se necesita un mecanismo que permita crear una base de datos de imágenes lo más homogénea posible de una zona urbana que sirva como base para geolocalizar una imagen determinada.

Google Maps es un servicio de Google lanzado en 2005 que ofrece mapas así como imágenes de satélite de prácticamente todo el mundo. En mayo de 2007 Google incorporó a Google Maps una nueva funcionalidad: Google Street View. Google Street View permite al usuario "navegar" por las calles de ciudades de todo el mundo ofreciendo panorámicas de 360º.

En un principio (2007), Google Street View sólo incluía 5 ciudades estadounidenses. Desde entonces no ha parado de incorporar nuevas ciudades y actualmente (2014), está presente en los cinco continentes. Es de destacar que tanto Europa y EEUU están digitalizados prácticamente al completo.

Las imágenes son obtenidas mediante vehículos que incorporan varias cámaras que capturan continuamente imágenes que son almacenadas junto con otros datos, como la ubicación GPS, velocidad y dirección del vehículo, ... Posteriormente estas imágenes son procesadas y convertidas es panorámicas de 360º. La visualización de estas imágenes en Internet es trivial. Sin embargo, su descarga, no lo es, ya que no está pensado para ello. De esta manera, existen dos principales formas de descargar imágenes de Google Street View:

2.6.1. API estática oficial de Google Street View

La API estática de imágenes de *Google Street View* [22] ofrece la posibilidad de obtener una imagen deseada, mediante distintos parámetros que se le pueden indicar en la siguiente url.

http://maps.googleapis.com/maps/api/streetview?parametros

Los parámetros van separados por el carácter *&* y son los siguientes:

- *size:* Tamaño de la imagen a obtener, hasta un máximo de 640 x 640 píxeles.
- *location:* Localización de la imagen. Puede ser una cadena de texto con la dirección de la imagen o las coordenadas en forma de *latitud, longitud.*

- *sensor (true, false)*: Para indicar si la petición procede de un dispositivo que use sensor de ubicación, con el fin de determinar la ubicación enviada en la solicitud

- *heading:* Indica el ángulo de la cámara (0-360)

- *fov:* Indica el zoom de la imagen. Valores pequeños indican valores más elevados de zoom.

- *pitch:* Inclinación del ángulo de la cámara hacia arriba o hacia abajo. Se expresa en grados.

Un ejemplo de conseguir una imagen de la fachada de la facultad de Informática utilizando esta API es la siguiente petición:

http://maps.googleapis.com/maps/api/streetview?size=640x640&location=Facultad%20de%20informatica,madrid&fov=90&heading=180&pitch=10&sensor=false

El resultado de esta petición se presenta en la Figura 2.11.

Fig. 2.11. Ejemplo de descarga con la API estática de Google Street View

A pesar de que el proceso de obtener una imagen es bastante sencillo, construir una base de datos que contenga todas las imágenes necesarias es complicado. Otro inconveniente que presentan las imágenes descargadas con esta API es que su resolución de 640 x 640 píxeles.

2.6.2. API estática no publicada de Google Street View

Esta API de Google Street View fue creada por Jamie Thompson en 2010 [23]. Con esta API se pueden obtener los "tiles" (baldosas) que componen las imágenes esféricas de Google Street View. Un ejemplo de cómo funciona se puede encontrar en [24], donde se pueden visualizar cada uno de los tiles que componen una imagen esférica.

El funcionamiento de esta API es el siguiente [25]:

http://cbk0.google.com/cbk?output=tile&panoid=[id]&zoom=[zoom]&x=[]&y=[]

- *output=tile*: Indica que lo que se quiere obtener es una baldosa
- *panoid:* El id de la imagen esférica de la que se quiere extraer los tiles.
- *zoom:* Nivel de zoom
- *x:* Determina la posición en el eje X de la cuadrícula de la imagen esférica. (Empieza en 0)
- *y:* Determina la posición en el eje Y de la cuadrícula de la imagen esférica. (Empieza en 0)

En la Figura 2.12 se puede ver una representación de la cuadrícula de una imagen esférica de Google Street View.

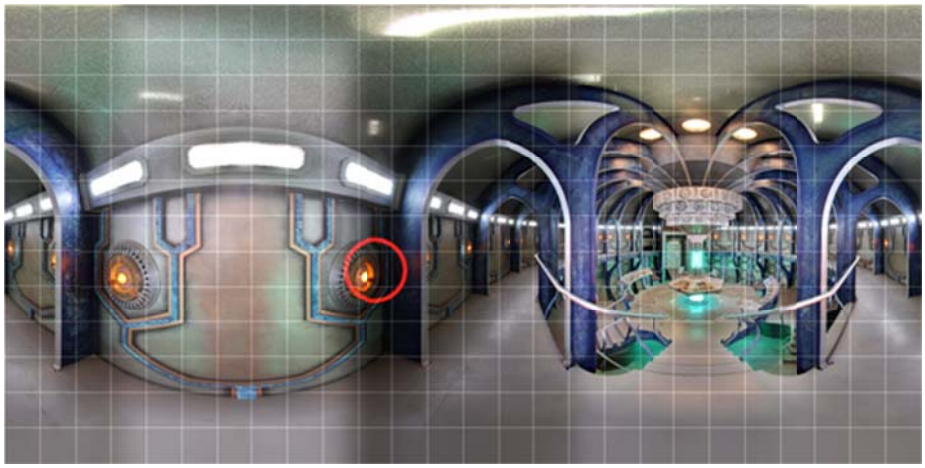


Fig. 2.12. Representación de la cuadrícula de una imagen esférica de Google Street View[25]

De esta manera si para obtener el *tile* o baldosa que está rodeado por un círculo rojo en la Figura 2.12, se escribiría la siguiente url:

http://cbk0.google.com/cbk?output=tile&panoid=lKxUOImSaCYAAAQIt71GFQ&zoom=5&x=10&y=7

Obteniendo como resultado la Figura 2.13.

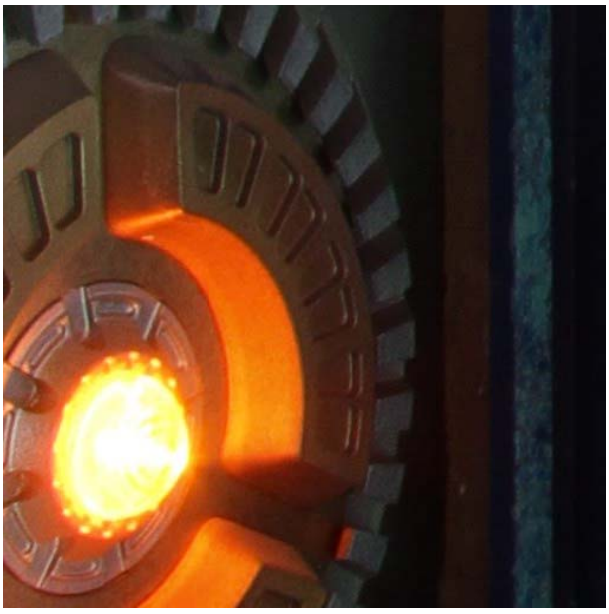


Fig. 2.13. Tile de una imagen esférica en su más alta resolución.

La ventaja de esta API frente a la API oficial, es que permite obtener imágenes de mucha mayor resolución.

3. USO DE STREET VIEW Y DESCRIPTORES EN EL PROCESO DE GEOLOCALIZACIÓN DE IMÁGENES DIGITALES

En [26] se propone un método para encontrar de forma automatizada los elementos arquitectónicos que son representativos de una ciudad dada una base de datos de una ciudad. Los elementos descubiertos, a través del algoritmo propuesto son, además de discriminantes geográficamente, significativos para las personas, haciéndolo útil para numerosas aplicaciones.

Se utilizó Google Street View para capturar las imágenes del área a estudiar y así crear la base de datos de imágenes de cada ciudad. Por cada panorámica, se extrajeron dos imágenes, una a cada lado del coche que toma las fotos, aproximadamente 10000 fotos por cada ciudad. Se utilizaron 12 ciudades para realizar los experimentos.

Para encontrar los elementos característicos de cada lugar, se buscan patrones que sean frecuentes en dicho lugar y que solo aparezcan en este lugar y en ningún otro. Por tanto, la base de datos se divide en dos partes:

- **Conjunto positivo**: Base de datos de la ciudad a la que se quiere encontrar sus elementos característicos
- **Conjunto negativo**: Base de datos del resto de ciudades.

El algoritmo utilizado es el siguiente:

1. Se comienza con un número determinado de parches candidatos aleatorios, y a cada parche se le da la oportunidad de ver si converge a un conjunto que es frecuente y discriminativo.
2. Para la descripción de cada parche se utilizaron los descriptores HOG más un descriptor de color.
3. Se procesan los vecinos de cada candidato y se rechazan aquellos candidatos que tienen muchos vecinos en el conjunto negativo.

4. Se construyen conjuntos aplicando aprendizaje discriminante iterativo a cada candidato.

La salida del algoritmo se muestra en la Figura 3.1, en la que se pueden observar elementos característicos de las calles de París, como pueden ser los rótulos de las calles, las farolas o las puertas principales de los edificios.

Fig. 3.1. Elementos característicos extraidos de París [26]

En [27] se trata de localizar imágenes usando como base de datos imágenes en 3D basado en la afirmación que un modelo 3D de imágenes aporta más información que es útil para conseguir una mayor precisión en la localización de una imagen. La construcción de modelos 3D del tamaño de una ciudad basados en millones de puntos es posible gracias a los avances en la investigación en SFM (Structure for Motion). El tamaño de estos modelos crea la necesidad de crear métodos que puedan manejar grandes bases de datos. El

trabajo se basa en mejorar la comparación de 2D a 3D ("2D to 3D matching) y en un framework para realizar esta tarea de forma eficiente, asociando puntos 3D a un vocabulario visual.

Para llevar a cabo la tarea de localizar una imagen, establecen correspondencias entre las características 2D de la imagen consultada y los puntos 3D del modelo. La forma habitual de hacerlo es comparando los descriptores SIFT de ambas imágenes (*Direct Matching*). Dado que esta búsqueda se vuelve inasumible cuando se tiene un gran número de descriptores, los tiempos de búsqueda se reducen usando "*indirect matching*".

Los experimentos se realizaron construyendo 3 bases de datos: base de datos de Roma y Dubroknik usando fotos de Flickr y una base de datos de Viena con imágenes obtenidas por una única cámara.

En [28] utiliza Google Street View como apoyo al desarrollo de los sistemas de asistencia a la conducción. El trabajo describe como acceder a la API de Google Street View para obtener la información necesaria para llevar a cabo sus diferentes aplicaciones. Por ejemplo, ponen el reconocimiento de señales usando las imágenes obtenidas de Google Street View. Así se indica que no está claro si se incumplen los términos de uso de la API de Google Street View aunque mencionan que Google concede acceso gratuito e ilimitado a universidades. Los datos que obtienen de la API de Google Street View son los siguientes:

- Thumbnail dadas unas coordinadas GPS: Dadas unas coordinadas GPS se puede obtener una imagen jpeg de 360º con un tamaño máximo de 300x128 píxeles.
- XML: Dadas unas coordinadas GPS se puede obtener información adicional de una imagen como su id o el tipo de calle.
- Tile: Dado un id se puede obtener una imagen indicando las coordenadas x e y, y un nivel de zoom

- Thumbnail dado un id: Dado un id devuelve una imagen de 360º, como en el primer punto.

En [29] se presenta un algoritmo que permite localizar una imagen usando Google Street View y localizar una imagen de mala calidad usando otras imágenes del mismo álbum. En este algoritmo, la imagen que se quiere localizar es comparada con una imagen geolocalizada de una base de datos. La localización de esta última es usada para encontrar la localización GPS de la imagen que buscamos. Como base de datos de referencia se utiliza Google Maps Street View. De las imágenes de Google Street View se extraen los descriptores SIFT organizados usando árboles y utilizan el método de Nearest Neighbor para recorrer los árboles. Se introduce el concepto de confianza de localización que se mide mediante la distribución de los votos de kurtosis. Así. cuando mayor es el valor de kurtosis mayor es el valor de la variable confianza de localización.

La base de datos utilizada en este trabajo está formada por imágenes tomadas de Google Street View. Google Street View toma una foto de 360º cada 12 metros. Las ventajas de GSV son la precisión (360º cada 12 m) y la uniformidad de la base de datos (la tarea de localizar es independiente de la popularidad del sitio). Aunque también presenta complicaciones, ya que la necesidad de capturar un gran número de imágenes y las limitaciones de almacenamiento hace necesario almacenar esas imágenes en una calidad inferior exigiendo que el algoritmo sea capaz de funcionar con una base de datos de imágenes de mala calidad. En los experimentos se utilizaron 100000 imágenes de GSV de Pittsburgh y Orlando. Las imágenes de esta base de datos son capturadas aproximadamente cada 12 metros y por cada punto se capturan 5 imágenes: las 4 de los lados y una de arriba.

El proceso para localizar una imagen sigue el siguiente:

1. Descargar las imágenes de la ciudad/región que queremos procedentes de

Google Street View.

2. Detectar los puntos de interés de las imágenes descargadas usando los descriptores SIFT.
3. Obtener los descriptores de los puntos de interés detectados, y se organizan en un árbol usando FLANN.
4. Extraer los puntos de interés y sus descriptores de la imagen de la que se desconoce sus coordenadas GPS.
5. Encontrar los NN de cada descriptor obtenido en la base de datos. Cada uno de los NN de los que se encuentra coincidencia en la base de datos, suma un voto para la imagen de la base de datos a la que corresponde.
6. Realizar una función de poda, con el fin de eliminar las coincidencias no deseadas. En este caso una función de poda toma más importancia ya que muchos de los descriptores procesados pertenecen a objetos no informativos como personas, coches, entre otros.
7. Selecciona la imagen con mayor número de votos.

En [30] se presenta un método que busca las vistas de una escena urbana a partir de una imagen digital utilizando un método basado en votos con descriptores SIFT. Para evaluar la efectividad del algoritmo realizaron experimentos con una base de datos de fotos urbanas de la ciudad de París, tomadas a mano. La propuesta forma parte del proyecto iTowns, en el que un usuario puede hacer una foto de un sitio, como un restaurante, por ejemplo, y la aplicación le responde con información acerca del sitio, en este caso del restaurante. Para conseguir la imagen en la base de datos que corresponde a la imagen objeto de investigación, se compara ésta con todas las imágenes de la base de datos. De esta forma, la imagen que obtenga más votos es la que corresponde con la imagen analizada. Sin embargo, al ser un algoritmo con coste lineal, es inasumible para grandes bases de datos. Por tanto, un método más efectivo es encontrar las mejores coincidencias comparando los puntos clave (keypoints) de la imagen consultada con los puntos clave de todas las

imágenes de la base de datos. El algoritmo utilizado es el siguiente:

1. Para cada punto de la imagen consultada, encontrar el vecino más cercano (k-NN) en la base de datos
2. Para cada vecino encontrado, añadir un voto a la correspondiente imagen.
3. Ordenar las imágenes por número de votos en orden descendiente.

La principal diferencia con el primer método es que en este caso cada punto clave de la imagen consultada tiene k coincidencias. En consecuencia, puntos de la imagen consultada que no tienen puntos correspondientes en la base de datos (puntos de objetos que no están en la base de datos, por ejemplo) seguirán votando. Estos votos son aleatoriamente distribuidos a lo largo de las imágenes y contribuyen a incrementar la posición en el ranking de imágenes irrelevantes.

Para eliminar la influencia de esos puntos irrelevantes se aplica una restricción geométrica a las coincidencias.

- **Búsqueda del vecino más cercano:** Hay varias técnicas para la búsqueda del vecino más cercano en grandes bases de datos. En este caso, los autores eligieron Multicurvas [31]. En particular, usaron Multicurvas con 4 curvas, cada una de ellas indexando 32 de las 128 dimensiones que componen el espacio de entrada SIFT.
- **Consistencia geométrica:** Para eliminar la influencia de los puntos irrelevantes, se estudian varios métodos: Diferencias de ángulos y Ransac [32].

4. DISEÑO E IMPLEMENTACIÓN DE UNA HERRAMIENTA DE GEOLOCALIZACIÓN DE IMÁGENES DIGITALES

En este capítulo se presentan todos los elementos relacionados con el ciclo de vida del proyecto a un alto nivel. Entre estos aspectos se debe resaltar la planificación y el seguimiento del mismo. En la sección 4.1 se presenta la planificación del proyecto. Las principales características se presentan en la sección 4.2. La sección 4.3 describe los diferentes módulos que conforman el proyecto. A continuación, se presenta el pseudocódigo del mismo en la sección 4.3. Las herramientas utilizadas en el desarrollo el proyecto se presentan en la sección 4.4 De igual forma, se presentan se realiza una clasificación de los medios y recursos que han sido necesarios en el transcurso del proyecto.

4.1. Planificación del Proyecto

El proyecto se ha desarrollado en 4 fases: Definición, Ejecución, Control y Documentación del Proyecto.

1. **Fase de definición del proyecto:** Esta fase tiene como objetivo la definición y delimitación del proyecto. En esta fase se realizaron varias reuniones con los tutores para exponer y enmarcar el proyecto de fin de grado, establecer las pautas del trabajo y definir horarios de tutorías y seguimiento del mismo. La fase tuvo una duración de 40 días e inició el 28 de octubre de 2013. Las actividades realizadas en esta fase se presentan en la Tabla 4.1.

Actividad	Duración (Días)	Comienzo	Fin
Reuniones con los tutores	10	lun 28/10/13	vie 08/11/13
Estudio del estado del arte	21	vie 08/11/13	vie 06/12/13
Definición del proyecto	10	lun 09/12/13	vie 20/12/13

Tabla 4.1. Actividades de la fase de definición del proyecto

2. **Fase de ejecución del proyecto:** Esta fase tiene como objetivo el desarrollo del proyecto. En esta fase se realizaron actividades de especificación de requisitos, diseño, implementación y experimentos. La fase tuvo una duración de 139 días e inició el 8 de enero de 2014. Las actividades realizadas en esta fase se presentan en la Tabla 4.2.

Actividad	Duración (Días)	Comienzo	Fin
Análisis y especificación de requisitos	**33**	**mié 08/01/14**	**vie 21/02/14**
• Investigación sobre características de imágenes	18	mié 08/01/14	vie 31/01/14
o Tipos de características	4	mié 08/01/14	lun 13/01/14
o Detectores de caracteristicas	4	mar 14/01/14	vie 17/01/14
o Descriptores de características	4	lun 20/01/14	jue 23/01/14
o SIFT	4	vie 24/01/14	mié 29/01/14
• Especificación de requisitos sobre descarga de imágenes	7	jue 30/01/14	vie 07/02/14
• Estudio y especificación de requisitos de extracción y comparación de descriptores de imágenes	10	lun 10/02/14	vie 21/02/14
Diseño del sistema	**30**	**lun 24/02/14**	**vie 04/04/14**
• Diseño de pantalla inicial de geolocalización	3	lun 24/02/14	mié 26/02/14
• Diseño de descarga y almacenamiento de dataset de imágenes	5	jue 27/02/14	mié 05/03/14
• Diseño de extracción y almacenamiento de descriptores de dataset de imágenes	6	jue 06/03/14	jue 13/03/14
• Diseño de comparación y filtrado de imágenes	12 días	vie 14/03/14	lun 31/03/14
• Diseño de presentación de resultados	4 días	mar 01/04/14	vie 04/04/14
Implementación del sistema	**69 días**	**lun 07/04/14**	**jue 10/07/14**
• Descarga de imágenes	16 días	lun 07/04/14	lun 28/04/14
• Obtención de coordenadas a descargar	7 días	mar 29/04/14	mié 07/05/14
• Extracción y almacenamiento de descriptores	9 días	jue 08/05/14	mar 20/05/14
• Comparación de imágenes	16 días	mié 21/05/14	mié 11/06/14
• Interfaz de resultados	4 días	jue 12/06/14	mar 17/06/14
Realización de pruebas	**15 días**	**vie 20/06/14**	**jue 10/07/14**

Tabla 4.2. Actividades de la fase de Ejecución del Proyecto.

3. **Fase de control:** Esta fase tiene como objetivo realizar actividades de seguimiento y control de todo el avance del proyecto, realizando los ajustes necesarios. La fase tuvo una duración de 139 días e inició el 8 de enero de 2014. Esta fase se realiza en paralelo a la fase de ejecución.

4. **Fase de documentación:** Esta fase tiene como objetivo generar toda la documentación del proyecto en las fases de Ejecución y control. La fase tuvo una duración de 176 días e inició el 8 de enero de 2014. Esta fase también se realiza en paralelo a la fase de ejecución y termina con la generación de la memoria final del proyecto y la presentación de la misma. Las actividades realizadas en esta fase se presentan en la Tabla 4.3.

Actividad	Duración (Días)	Comienzo	Fin
Generación de documentación del proyecto	139 días	mié 08/01/14	lun 21/07/14
Redacción de la memoria	25 días	mar 22/07/14	lun 25/08/14
Preparación de la presentación	13 días	lun 25/08/14	mié 10/09/14

Tabla 4.3. Actividades de la fase de documentación del proyecto

La Figura 4.1 muestra la planificación se ha realizado utilizando diagramas de Gantt.

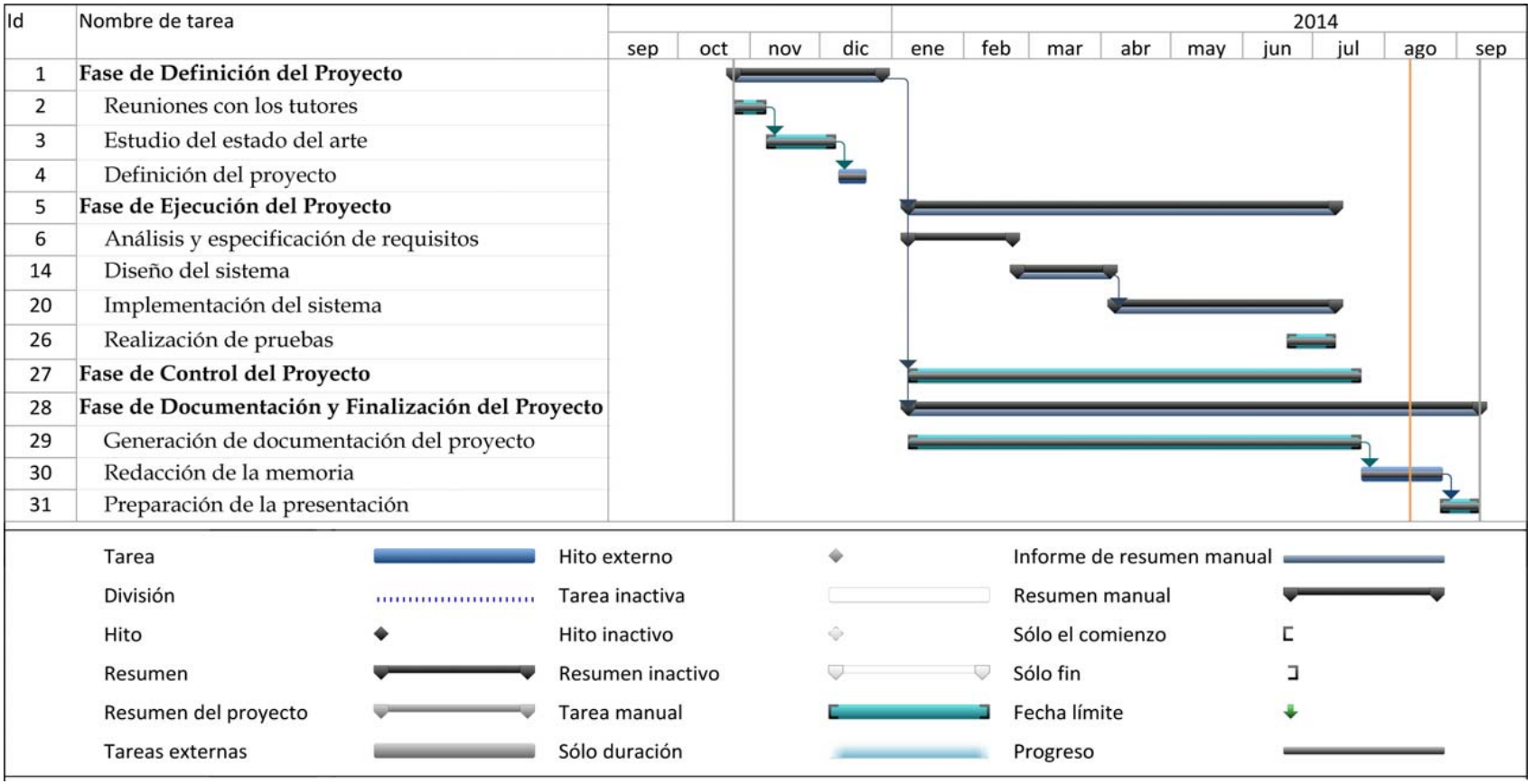


Fig. 4.1. Diagrama de Gant de la planificación del proyecto

4.2. Generalidades

La aplicación diseñada para este trabajo se divide en 3 grandes procesos:

a. Descarga de imágenes: En un primer lugar se idéntica la zona en la que podría estar la imagen. Para ello se indica, a través de Google Maps, los puntos del mapa de los que se quieren descargar las imágenes para componer la base de datos. A continuación, se procede a descargar las imágenes a través de Matlab.

b. Extracción de descriptores: Una vez descargadas todas las imágenes que forman la base de datos, se procede a extraer sus puntos clave y sus correspondientes descriptores, almacenando dicha información en disco.

c. Comparación de descriptores: Por último, se extraen los descriptores de la imagen a consultar y se comparan sus descriptores con los de todas las imágenes de la base de datos.

4.3. Módulos del Proyecto

En esta sección se presenta la estructura principal de la aplicación desarrollada, mostrando de forma específica el diseño y la implementación de los módulos que la conforman.

4.3.1. Descarga de Imágenes

Una parte imprescindible de este trabajo es la obtención de las imágenes que compondrán nuestra base de datos. Como se indica, se pretende utilizar el servicio "*Street View*" de Google, puesto que compone la mayor base de datos de escenas urbanas.

Como se explica en el apartado 2.1 la API no oficial de Google Street View permite obtener imágenes de mayor resolución, lo que para este trabajo es de especial importancia.

De esta manera, en una segunda fase de investigación, se encontró el trabajo "*Building Streetview Datasets for Place Recognition and City Reconstruction*" [33] que facilita la descarga de imágenes de *GSV* con el fin de construir bases de datos que sirvan para el reconocimiento de lugares y la reconstrucción de ciudades y que precisamente utiliza esta API.

Para construir la base de datos, divide el proceso en dos partes:

En primer lugar obtiene los datos necesarios de las imágenes que quiere descargar, es decir, las id de las imágenes de la ciudad o ciudades de las que se quiere construir la base de datos. Para ello utiliza la API javascript de Google Maps creando un html, en el que se le especifica en el código las coordenadas de la ciudad a descargar y al ejecutarse procede a obtener los id de las imágenes deseados.

Una vez obtenida esta información, en forma de datos con los que el usuario se crea dos ficheros de texto, se ejecuta un código Matlab que es el encargado de descargar las imágenes individuales (miles de pequeños cuadrados) y ensamblarlos.

De esta manera el código Matlab genera dos tipos de imágenes:

- *cutouts:* imágenes en paralelo al coche de Google
- *Perspective view images*: Imágenes esféricas tal y como son generadas para la aplicación de *Google Street View.*

Las imágenes esféricas tienen una calidad enorme: 6656x3328 píxeles cubriendo una gran área. Sin embargo este tamaño presenta inconvenientes en cuanto a rendimiento.

Los *cutouts* son imágenes de una calidad aceptable y una resolución más asumible (936x537 píxeles). Por cada punto se descargan dos imágenes, una al lado izquierdo y otra del lado derecho lo que permite reconstruir una calle de manera bastante sencilla.

El gran inconveniente de este trabajo, es que descarga imágenes de la ciudad indicada, al azar. De manera que es necesario miles de imágenes para reconstruir una ciudad completa, y el coste se vuelve inasumible para un trabajo como el que se quiere llevar a cabo.

4.3.1.1. Identificación de la Zona

El objetivo inicial, es construir una base de datos que conste de unas pocas calles. Para ello se debe de modificar el código del HTML para obtener los id de las imágenes que queremos descargar.

La manera más sencilla es dando coordenadas a mano de todos puntos que se quieren descargar. Sin embargo, es bastante laborioso. Así, se procedió a diseñar un algoritmo que permita al usuario obtener estas coordenadas de manera visual. Se optó por utilizar la API javascript de Google Maps, de tal manera que el usuario pueda hacer clic en el mapa dando puntos, formando el recorrido que se desea descargar. Internamente, el algoritmo se encarga de interpolar los puntos intermedios en línea recta.

En la Figura 4.2 se puede apreciar como el usuario ha dado 7 puntos (marcadores). El código se encarga de interpolar los puntos intermedios entre los marcadores para conseguir que se descarguen imágenes cada aproximadamente 10 metros.

Fig. 4.2. Ejemplo de obtención de las coordenadas a descargar

El número de puntos intermedios vendrá dado en función de la distancia entre los dos puntos. Puesto que el objetivo es conseguir una imagen cada unos 10 metros, se divide la distancia de los dos puntos entre la distancia que define 10 metros. De esta manera se tiene:

$$Dist. = \sqrt{(lat_1 - lat_0)^2 + (lon_1 - lon_0)^2}$$

El número de puntos intermedios (num.puntos) viene dado por:

$$nº\ puntos = \frac{dist.entrePuntos}{dist.10metros}$$

Así la expresión en función de i para obtener los puntos intermedios queda de la siguiente manera:

$$(lat_i, lon_i) = (lat_0 + i \times \frac{lat_1 - lat_0}{nº\ puntos}), (lon_1 + i \times \frac{lon_1 - lon_0}{nº\ puntos}) i = 1,...nº\ puntos$$

De esta manera se transformó el HTML del trabajo de Gronat casi por totalidad, utilizando únicamente las funciones de obtención de los ids

correspondientes a las imágenes de Google Street View que se quieren obtener. Así, se introdujeron las siguientes mejoras (Ver Figura 4.3):

- Creación del algoritmo previamente explicado, para que el usuario pueda especificar de forma interactiva que región quiere descargar. Cada vez que el usuario elige un punto se llama a una función(obtienePuntos) que se encarga de realizar la interpolación de los puntos intermedios. Los puntos se almacenan en un array de puntos declarado como variable global.
- Incorporación de una caja de texto en la que el usuario pueda especificar el nombre de la descarga.
- Incorporación de una caja de texto sobre el mapa que permite al usuario buscar y posicionarse en la zona o lugar que desea descargar.
- Incorporación de un botón con el texto "Obtener ficheros". Cuando el usuario hace clic sobre este botón se llama a una función, de creación propia, se encarga de obtener las id de las imágenes correspondientes de cada punto, con la función ya creada de Gronat, y se encarga de escribir en dos variables de tipo String, el contenido de los dos ficheros que se pretenden descargar.
- Incorporación de una barra de progreso que ofrece al usuario feedback de cómo va avanzando el proceso anterior ("Obtener ficheros")
- Incorporación de dos botones con el texto "Descargar mapping.txt" y "Descargar download.txt". Estos botones están inicialmente ocultos y se muestran cuando el usuario hace clic sobre el botón "Obtener Ficheros". Una vez que el usuario hace clic sobre estos botones se llama a las funciones correspondientes, que haciendo uso de *FileSaver.js* [34], permite al usuario descargar los ficheros mapping.txt y download.txt.
- Incorporación de instrucciones para el usuario.

Fig. 4.3. Vista del HTML modificado, con todas las mejoras introducidas

4.3.1.2. Descarga de Imágenes

Una vez obtenidos los ficheros necesarios se puede proceder a la descarga de las imágenes. Antes de nada, se ejecuta una función que lee del fichero de configuración donde ha descargado los ficheros el usuario y los copia al directorio de trabajo. Una vez hecho esto, es necesario ejecutar Matlab. Puesto que la aplicación está programada en Python, era necesario encontrar la forma de ejecutar el programa Matlab desde Python. Para ello se valoraron varias posibilidades, optando finalmente por ejecutar Matlab desde línea de comandos, sabiendo que desde Python se pueden ejecutar comandos con el módulo Subprocess [35]. De esta manera, en la llamada a Matlab se le añade el parámetro *–wait* para que se bloquee la ejecución de la aplicación principal hasta que el programa de Matlab termina de descargar todas las imágenes.

El nombre con el que se guardan las imágenes es su posición (latitud y longitud), con lo que a la hora de realizar la comparación se sepa cual es la posición de la imagen a consultar.

4.3.2. Extracción de Descriptores

Para la extracción de puntos clave y descriptores se usó la librería OpenCV que proporciona una función que dada una imagen previamente cargada en Python, obtiene sus puntos clave y sus correspondientes descriptores.

En primer lugar se procede a cargar todas las imágenes de la carpeta en la que se han descargado en un array. Seguidamente se procede a recorrer dicho array, con el fin de obtener los puntos clave y los descriptores de cada foto.

Dado que esta operación tiene un coste computacional y en tiempo importante, no tenía sentido extraer los descriptores de todas las imágenes de la base de datos por cada vez que se quisiera consultar una imagen, así que se decidió guardar los puntos clave y los descriptores en disco.

Para guardar los puntos clave y los descriptores en disco se decidió utilizar el módulo de Python "*cPickle*" [36] que permite la serialización de objetos, es decir, transformar el estado de un objeto en un formato que se pueda almacenar, recuperar y transportar. Sin embargo, *cPickle* no permite guardar el objeto *cv2.KeyPoint* de OpenCV, por lo que se diseñó un método que guarda los atributos de cada punto clave en un array, junto con su descriptor y el nombre de la foto. Así , se construye un array cuyos elementos son arrays que contienen toda la información de cada foto (sus puntos clave, descriptores y nombre de la foto).

Finalmente se guarda el array en disco, indicando que guarde un archivo binario, (lo más eficiente).

4.3.3. Comparación de descriptores

Una vez obtenida las imágenes y la base de datos de descriptores se puede proceder a la comparación de la imagen que se quiere geolocalizar con todas las imágenes de la base de datos

Los pasos que se siguen en este proceso, muy abstraído, son los siguientes:

1. Se extraen los puntos clave y los descriptores de la imagen a consultar.
2. Se cargan los puntos clave y los descriptores de la base de datos de imágenes de Google Street View previamente obtenidos.
3. Se comparan los descriptores de la imagen a consultar con los descriptores de cada una de las imágenes de la base de datos.
4. Se ordenan los resultados de mayor a menor número de coincidencias.
5. Se presentan los resultados.

4.3.3.1. Extracción de puntos clave y descriptores de la imagen a consultar

En este paso se carga la imagen que ha seleccionado el usuario en una variable. A continuación se extraen los puntos clave y descriptores de dicha imagen, haciendo uso de la función correspondiente de OpenCV.

4.3.3.2. Carga de la base de datos de descriptores y puntos clave

En este paso se procede a cargar la base de datos de descriptores y puntos claves previamente creada. Para ello se diseñó un método que abre el archivo binario almacenado en disco y recupera el array guardado. Seguidamente se desempaqueta dicho array recorriéndolo, y cargando en listas los puntos clave, descriptores y los nombres de cada foto.

4.3.3.3. Comparación

En este paso se procede a realizar la comparación de la imagen a consultar con los descriptores de cada una de las imágenes de la base de datos.

En primer lugar se crea un comparador basado en FLANN [37]. FLANN es una librería que facilita las búsquedas del vecino más cercano en altos espacios dimensionales, como en este caso (un descriptor tiene 128 dimensiones). De esta manera proporciona una seria de algoritmos para llevar a cabo de la forma más eficiente este tipo de búsqueda. En este caso, el vecino más cercano se define como el punto clave cuyo descriptor tiene la menor distancia euclidea.

Al comparador le indicamos que use árboles kd [38] como algoritmo de indexación de los vecinos más cercanos. Un árbol kd es una estructura para almacenar un conjunto de puntos en un espacio dimensional k.

Seguidamente se procede a obtener los mejores resultados. Para ello se utiliza el comparador, indicando k = 2, como en el paper de Lowe [16], con el fin de obtener los dos vecinos más cercanos por cada punto clave.

A continuación se procede a realizar un filtrado con el fin de obtener solo las buenas coincidencias y descartar los posibles falsos positivos. Para ello se define un umbral (menor que 1): si la distancia euclidea de la primera coincidencia es menor que la distancia euclidea de la segunda coincidencia multiplicada por dicho umbral, estamos seguros de que es una buena coincidencia, ya que el segundo vecino más cercano está lo suficientemente lejos. En este trabajo se utiliza un umbral de 0.6 frente al 0.8 del paper de Lowe, con el fin de obtener menos falsos positivos.

De esta forma se recorren todas las coincidencias, realizando la comprobación del umbral para realizar el filtrado, y almacenando el número final de coincidencias en una variable contador.

Finalmente se crea un nuevo objeto de la clase *ImagenConCoincidencias* que contiene toda la información de la comparación de esa imagen de la base de datos con la de que se pretende obtener la localización:

- Nombre de la imagen de la base de datos
- Lista de puntos clave
- Lista de coincidencias
- Parámetros de dibujo
- Número de coincidencias.

Este objeto se almacena en una lista que contiene toda la información de las comparaciones realizadas. Al finalizar todas las comparaciones se procede a

ordenar dicha lista por el número de coincidencias, de mayor número a menor.

4.3.3.4. Resultados

Al usuario se le presentan los 10 mejores resultados obtenidos en la comparación. Por cada uno de ellos se presenta:

- Número de coincidencias con la imagen a consultar.
- Coordenadas de la imagen de Google Street View (latitud, longitud).
- Imagen de Google Street View.
- Imagen comparativa de la imagen a consultar con la de Google Street View

Aparte de toda la información anterior, al usuario se le ofrece la posibilidad de abrir la localización de la imagen en Google Maps en el navegador.

Para obtener dicha información se diseñó una función que recibe como parámetros la lista con la información de todas las comparaciones realizadas y tanto la imagen a geolocalizar como sus puntos clave.

La función recorre las 10 mejores coincidencias de la lista. Por cada una de ellas:

1. Se obtiene la imagen de la comparación entre ambas imágenes. Es decir, una imagen en la aparecen las dos imágenes contiguas con los puntos clave marcados con círculos rojos y las correspondencias representadas con líneas verdes entre dichos puntos clave. Un ejemplo se puede ver en la Figura 4.4.

2. Se guarda la imagen de Google Street View en una carpeta de resultados, renombrada con el número correspondiente al resultado (*1.jpg* para la imagen con más coincidencias, *2.jpg* para la siguiente, ...)

3. Se obtiene la latitud y longitud de la imagen de Google Street View. Como se indica en el punto 4.2.2, el nombre de cada imagen de las descargadas de Google Street View está formado por la latitud y la longitud, por lo que

basta con leer el nombre de la foto y separar la cadena por el carácter "_" para obtener los datos deseados.

4. Se guarda el número de coincidencias.

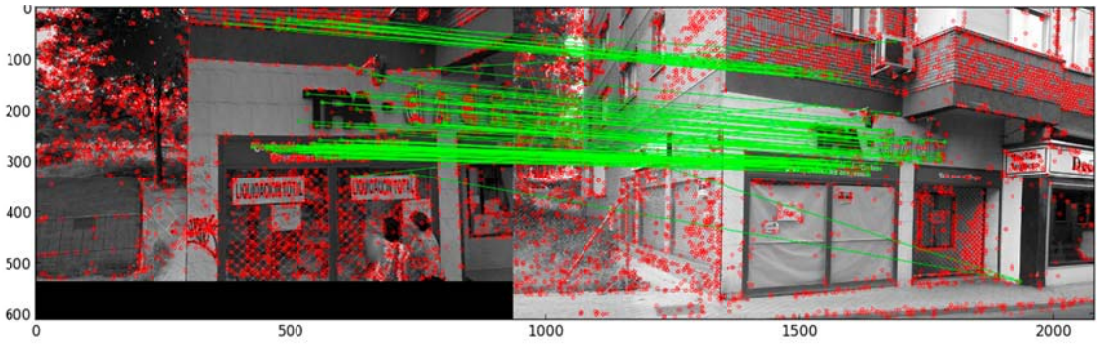


Fig. 4.4 Ejemplo de imagen de comparación.

Con toda esta información se construye una lista que se pasa como parámetro a la ventana de resultados.

4.4. Pseudo-Código

El pseudo-código del programa principal es el siguiente:

1. Seleccionar la imagen que se desea geolocalizar.
2. Identificar la región y descargar imágenes de Google Street View.
3. Extraer los descriptores.
4. Comparar la imagen que se desea geolocalizar con las imágenes de la base de datos.
5. Presentar los mejores resultados.

El proceso de delimitación de la zona y descarga de las imágenes desde Google Street View es el siguiente:

1. El usuario selecciona el punto inicial en el mapa de Google Maps.
2. El usuario selecciona un segundo punto y se interpolan los puntos intermedios en línea recta.

3. El usuario continua seleccionando puntos en el mapa (obteniéndose por interpolación los puntos intermedios), hasta que completa la zona de la que pretende descargar las imágenes.
4. Se obtienen los identificadores de las imágenes de Google Street View
5. El usuario se descarga los ficheros con la información necesaria para descargar las imágenes.
6. Se descargan las baldosas (tiles) que componen cada imagen esférica de Street View.
7. Se componen las imágenes esféricas
8. Se generan las imágenes en paralelo a las fachadas de los edificios que se utilizan en el trabajo.

El proceso de Extracción de descriptores es el siguiente:

1. Se carga cada una de las imágenes de la base de datos
2. Se extraen los puntos clave y los descriptores de cada una de las imágenes
3. Se guarda dicha información en un array.
4. Una vez extraídos todos los puntos clave y los descriptores de todas las imágenes de la base de datos, se guarda el array en un archivo binario en disco.

El proceso de comparación de la imagen objeto de investigación con las imágenes de la base de datos:

1. Se carga la imagen que se desea geolocalizar y se extraen sus puntos clave y sus descriptores.
2. Se abre el archivo binario que contiene el array con los puntos clave y descriptores de la base de datos.
3. Se comparan los descriptores de la imagen que se pretende geolocalizar con los descriptores de cada una de las imágenes de la base de datos

4.5. Herramientas de Desarrollo

Para el desarrollo del proyecto se utilizaron los siguientes lenguajes de programación: python, HTML + javascript, Matlab.

4.5.1. Python

Se ha utilizado el lenguaje de programación Python en su versión 2.7.6 ya que tiene una licencia de código abierto (*Python Software Foundation License).* Esta licencia es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1.

Se han utilizado las siguientes librerías:

- **NUMPY**: Librería para la realización de cálculos matemáticos, y proporciona otras funciones muy útiles como la transformada discreta de Fourier.

- **MATPLOTLIB**: Es una biblioteca para la generación de gráficos a partir de datos contenidos en listas o vectores.

- **OPENCV**: Librería de código abierto que proporciona una infraestructura común para aplicaciones de visión por computador. La versión utilizada es 3.0, aunque esta versión todavía en desarrollo, proporciona nuevas funciones indispensables para este trabajo.

- **CPICKLE:** Módulo que permite la serialización de objetos, para, por ejemplo, guardarlos en disco.

- **WEBBROWSER**: Módulo que permite la apertura en el navegador de páginas web.

- **SHUTIL**: Módulo que permite operaciones de alto nivel con ficheros, como copia, borrado, ...

- **SUBPROCESS:** Módulo que permite ejecutar procesos por línea de comandos.

4.5.2. Wget

Para la descarga de imágenes de Google Street View, se hace uso del comando wget. Para ello es necesario descargar la versión para Windows, desde la página del proyecto GnuWin32.

4.5.3. Matlab

Se utiliza para descargar las imágenes de Google Street View. Se utilizó la versión del año 2013.

5. EXPERIMENTOS Y RESULTADOS

Este capítulo presenta los experimentos realizados para evaluar la herramienta y analizar los resultados obtenidos. Se detallan los experimentos desde las pruebas iniciales hasta los resultados utilizando la aplicación.

5.1. Pruebas Iniciales

Una vez desarrollado el algoritmo de comparación de dos imágenes utilizando la extracción de descriptores, se procedió a la realización de pruebas.

Se realizó una prueba lo más sencilla posible, consistente en realizar una foto a un objeto, solo y después tomar una foto en una escena en la que aparezca dicho objeto junto a otros. El resultado de la prueba se presenta en la Figura 5.1.

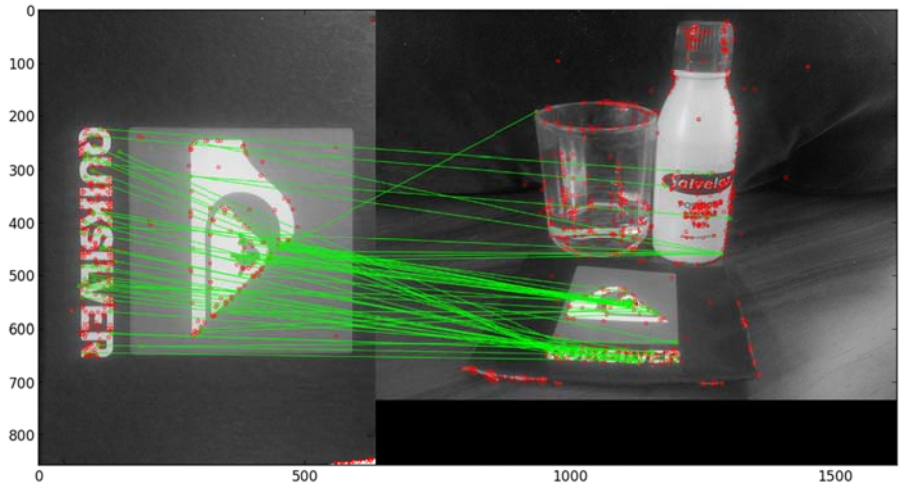


Fig 5.1. Comparación de la imagen de un objeto con otra escena que contiene varios objetos.

A la izquierda de la Figura 5.1 se tiene el objeto en solitario y a la derecha aparece una escena compuesta por dicho objeto y otros objetos más. Los puntos rojos representan los puntos clave y las líneas verdes correspondencias entre los

descriptores de una imagen con otra. De esta manera se puede comprobar cómo, a pesar de que en la escena el objeto aparece rotado y en distinta perspectiva, el código es capaz de detectar un gran número de correspondencias, aunque también aparecen algunos falsos positivos (líneas verdes que van al vaso y el bote). Como se comentará más adelante el filtrado de estos falsos positivos es fundamental para conseguir buenos resultados.

Tras este primer contacto con la comparación de imágenes, se realizaron pruebas con escenas urbanas. Para ello se recorrieron varias calles andando, para tomar fotografías con el teléfono móvil, en este caso, un Samsung Galaxy SIII, con una cámara de 8 megapíxeles, con el fin de comparar dichas fotografías con las descargadas de Google Street View.

Las calles fotografiadas fueron las siguientes:

- Calle Isla de Oza
- Calle Aurora
- Calle Isla de Nelson
- Avenida Complutense

5.2. Pruebas individuales

En un primer lugar se realizaron comparaciones de las imágenes seleccionando a mano la imagen de Google Street View correspondiente a la tomada con el teléfono móvil. De esta manera, se realizaron 18 pruebas, haciendo uso de descriptores SIFT y SURF con el fin de descubrir con cual de ellos se obtienen mejores resultados.

Puesto que las imágenes tomadas con el teléfono móvil tienen mucha resolución, se procedió a reducirlas de resolución y comprimirlas con jpeg, con el fin de que la comparación se realizara de manera más rápida y pareja. Así las imágenes pasaron de ocupar unos 3 MB a 200 kb.

Puesto que muchas de las pruebas son muy similares entre si, a continuación se muestran 6 pruebas que cubren la mayoría de los distintas escenas urbanas que se pueden encontrar. En la Tabla 5.1 se resumen dichas pruebas, indicando en cada una de ellas, el tiempo aproximado de ejecución del algoritmo y el número de coincidencias obtenidas entre las imágenes utilizadas.

Número	Calle	T. SIFT	T. SURF	Núm. Coincidencias SIFT	Núm. Coincidencias SURF
1	Isla de Oza	10	10	135	66
2	Isla de Oza	8	7	98	98
3	Isla de Oza	10	8	6	1
4	Calle Mayor	3	2	14	21
5	Av. Complutense	30	23	0	0
6	Calle Aurora	10	15	5	0

Tabla 5.1. Resultados tras la realización de pruebas individuales.

5.2.1. Isla de Oza I

La primera prueba se trata de una tienda. Como se puede observar en la Figura 5.2, las imágenes están tomadas desde una perspectiva ligeramente diferente. Otro aspecto a destacar es que las imágenes están tomadas en fechas diferentes, lo que en este caso se traduce en que el escaparate de la tienda es distinto (en la de Google Street View la tienda está en liquidación total y en la tomada con el teléfono la tienda ya ha cerrado). Este es un factor muy a tener en cuenta, ya que la diferencia entre la fechas en la que se tomaron las imágenes puede hacer que la geolocalización no tenga éxito, si la escena urbana ha cambiado significativamente en ese intervalo de tiempo.

Fig. 5.2. Foto usada en prueba Isla Oza I.

En la Figura 5.3 se puede observar los resultados de la comparación tras aplicar el descriptor SIFT. La mayoría de coincidencias detectadas pertenecen tanto al rótulo de la tienda como las letras situadas en la parte superior del escaparate, ya que son muy descriptivas. Otras coincidencias detectadas pertenecen a la pared del edificio. Se puede observar como aparecen unos cuantos falsos positivos (líneas verdes que unen puntos que no coinciden entre si).

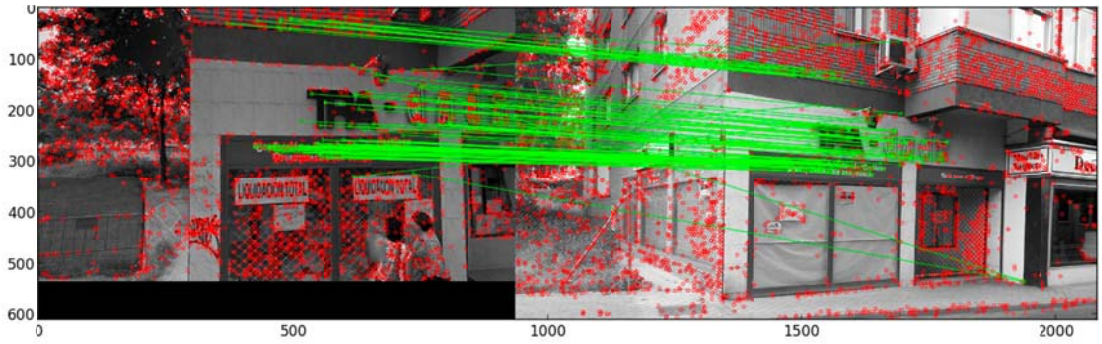


Fig. 5.3. Resultados de Prueba Isla Oza I con SIFT.

En la Figura 5.4 se muestran los resultados con SURF, muy similares a los del SIFT, aunque se obtienen menos coincidencias (66 frente las 135 de SIFT).

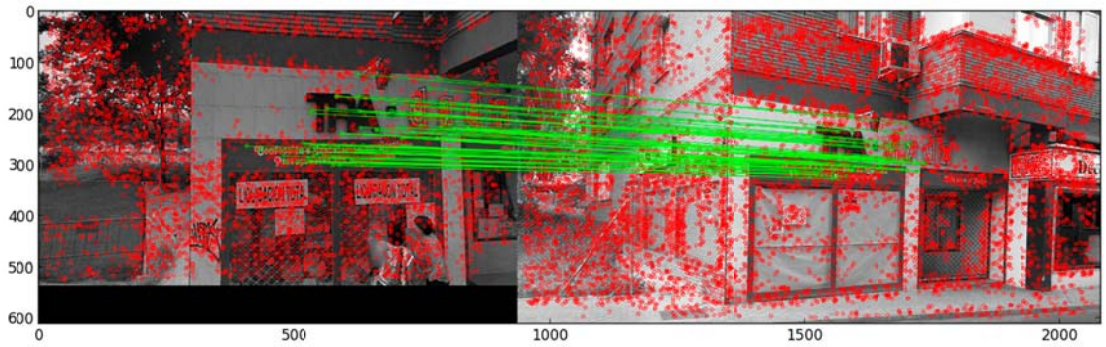


Fig. 5.4. Resultados de Prueba Isla Oza I con SURF

5.2.2. Isla de Oza II

La segunda prueba se trata de otro comercio, en este caso un taller, como se puede ver en la Figura 5.5. La escena es bastante similar, difiriendo en detalles como los grafitis, que han sido eliminados en la imagen tomada con el teléfono o el coche que está en reparación.

Fig. 5.5. Foto usada en prueba Isla Oza II.

Como se puede ver en la Figura 5.6 los resultados con SIFT son muy buenos. Como en el caso anterior, la mayoría de las coincidencias aparecen en el rótulo del taller. También aparecen coincidencias en las chapas de alarma y de vado. En este caso, aparecen algún falso positivo más.

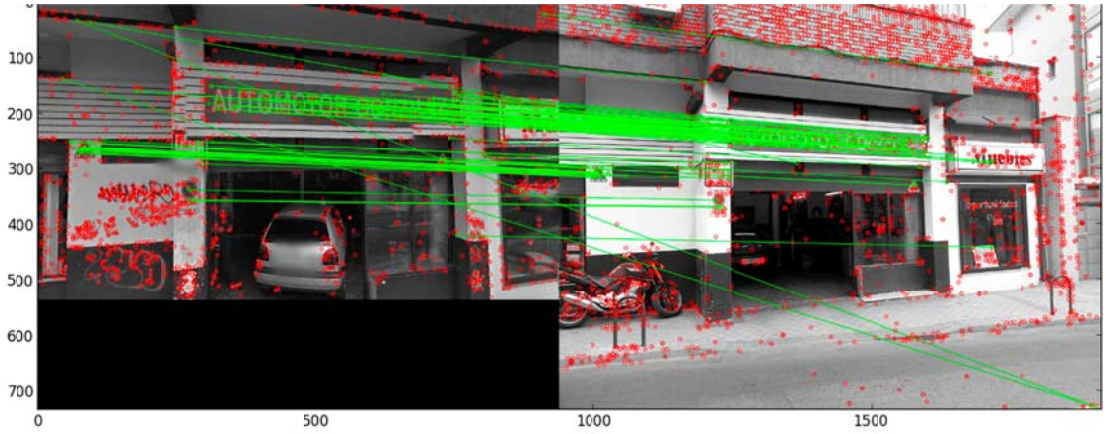


Fig. 5.6. Resultados de prueba Isla Oza II con SIFT

En la Figura 5.7 se muestran los resultados con SURF. Estos son muy similares a los obtenidos con SIFT y aunque se obtienen el mismo número de coincidencias, los resultados son menos consistentes (más falsos positivos).

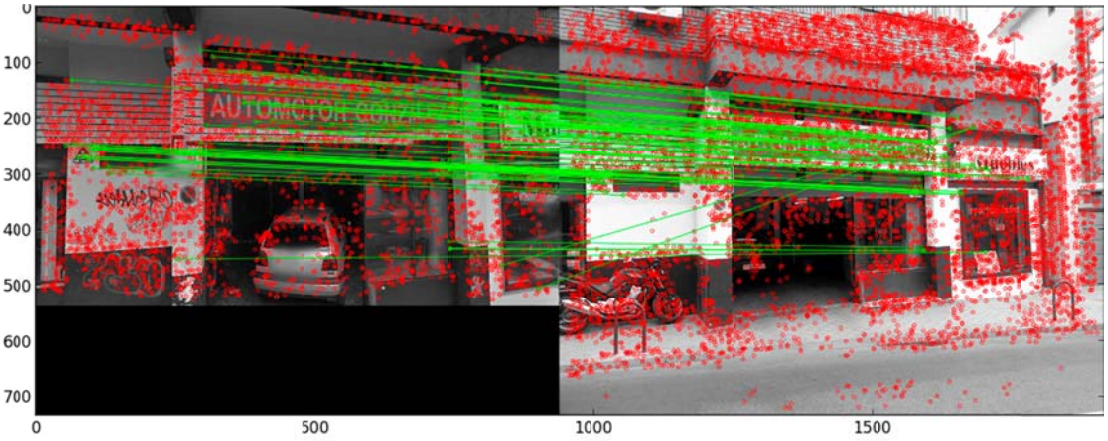


Fig. 5.7. Resultados de prueba Isla Oza II con SURF

5.2.3. Isla de Oza (III)

En la Figura 5.8 se presenta un escenario distinto a los dos anteriores. Las imágenes tienen la misma perspectiva y el mismo nivel de zoom, se observa la presencia de vegetación, un elemento que varia con facilidad a lo largo del tiempo. Además observamos que la escena está bastante más alejada que en los dos casos anteriores.

Fig. 5.8. Foto usada en prueba Isla Oza III.

En la Figura 5.9 se puede ver que los resultados, tras aplicar SIFT, son bastante negativos, ya que se obtienen pocas coincidencias (6) y además son todas falsos positivos. Las causas de este resultado vienen por lo poco descriptivo de la escena, el nivel de zoom (que hace que el nivel de detalle no sea el mejor.

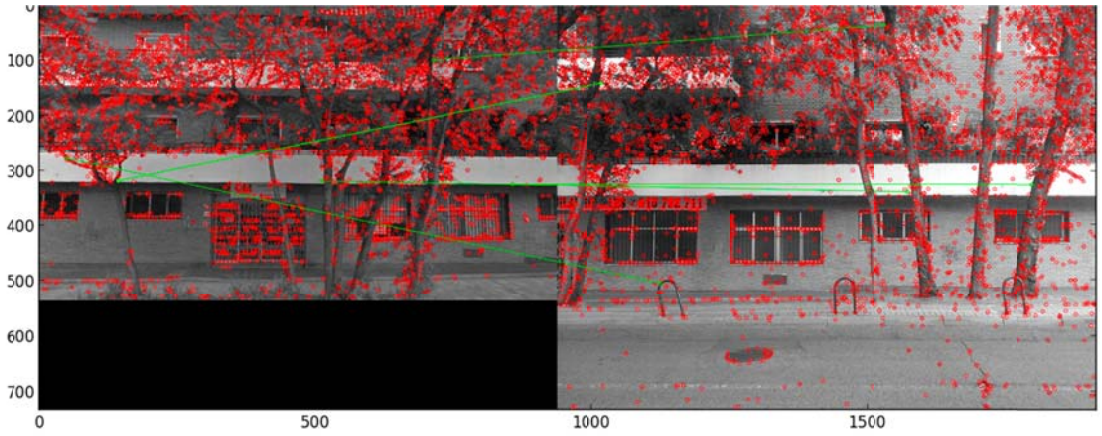


Fig. 5.9. Resultados de prueba Isla Oza III con SIFT

En la Figura 5.10 se muestran los resultados con SURF, en los que solo se obtiene una coincidencia, aunque es correcta.

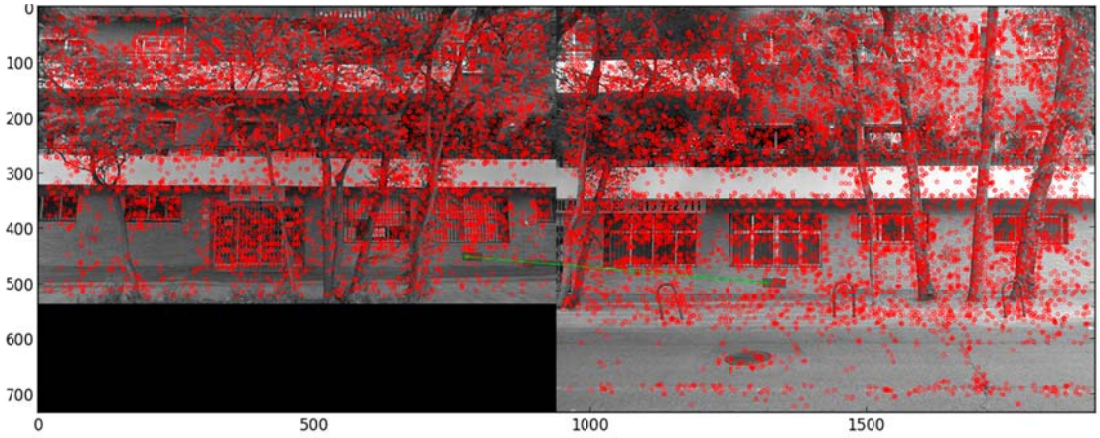


Fig. 5.10. Resultados de prueba Isla Oza III con SURF

5.2.4. Calle Mayor I

La Figura 5.11 muestra la cuarta prueba, en la que se presenta un escenario urbano distinto: centro histórico de Madrid. En concreto se trata de una fotografía tomada en la Calle Mayor. En este caso la imagen a comparar con la descargada de Google Street View se obtuvo de Internet. En concreto, se trata de parte de una fachada, en la que se observa una ventana y una placa con la inscripción "Aquí vivió y murió D. Pedro Calderón de la Barca". A la izquierda se presenta la imagen obtenida de Internet y a la derecha la imagen correspondiente descargada de Google Street View. Como se puede observar la placa está a punto de no aparecer, ya que las imágenes obtenidas del Street View no cubren todo el edificio.

Fig. 5.11. Foto usada en prueba Calle Mayor I.

Como se puede observar en la Figura 5.12 los resultados con SIFT son muy buenos, ofreciendo un buen número de coincidencias (14), con un número muy reducido de falsos positivos.

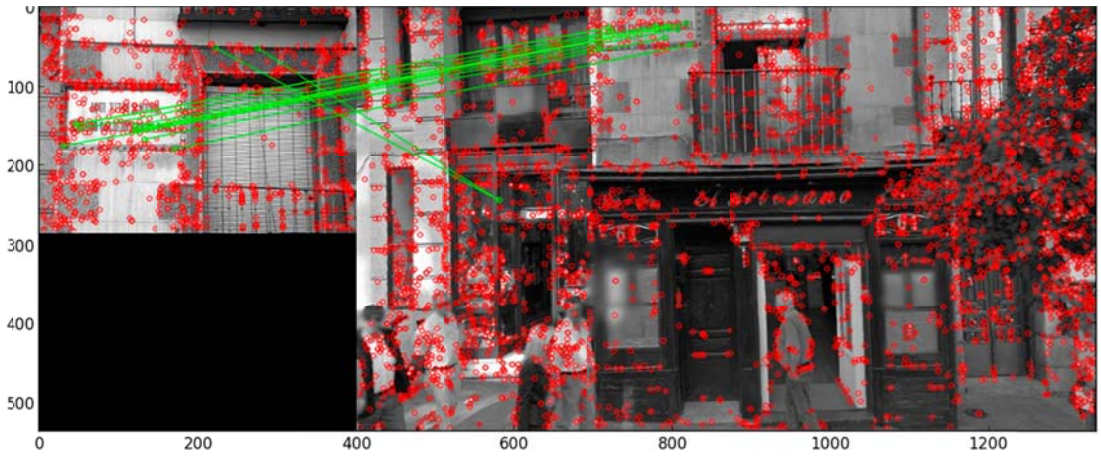


Fig. 5.12. Resultados de prueba calle Mayor I con SIFT

En la Figura 5.13 se muestran os resultados con SURF, muy similares a los obtenidos con SIFT ofreciendo alguna coincidencia más y con prácticamente ningún falso positivo.

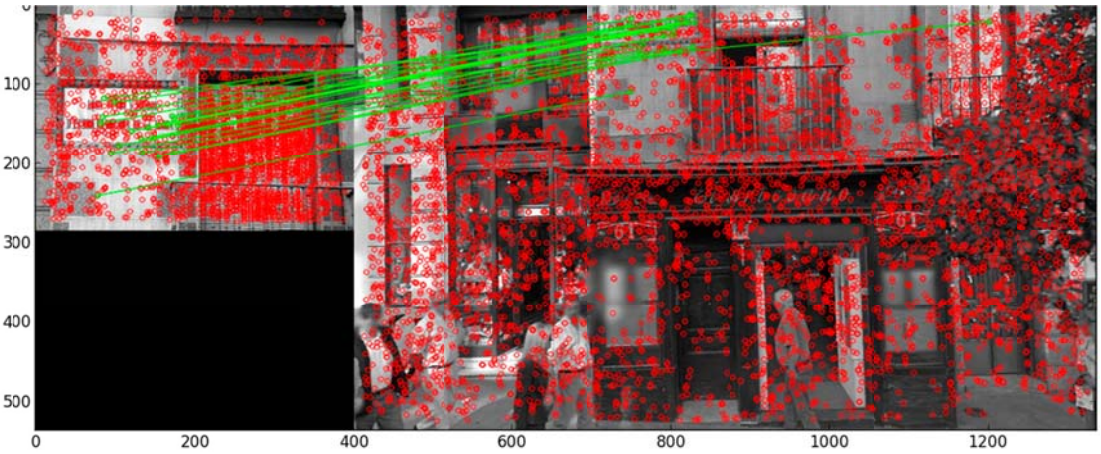


Fig. 5.13. Resultados de prueba calle Mayor I con SURF.

5.2.5. Avenida Complutense

En este prueba se tiene una imagen tomada en avenida Complutense, como se puede observar en la Figura 5.14. En concreto, se trata de una parte del vallado del Ciemat. Como se puede observar la escena consta de mucha

vegetación (césped y arboles), y de pocos elementos descriptivos, tanto la valla como la acera no ofrecen mucha información significativa. Además en este caso observamos como la perspectiva en las dos imágenes son distintas.

Fig. 5.14. Foto usada en prueba avenida Complutense.

La Figura 5.15 muestra los resultados obtenidos. El tiempo de la ejecución con SIFT es muy elevado, ya que el detector es muy sensible a la vegetación, obteniendo gran cantidad de puntos clave (puntos rojos). Sin embargo, no se obtienen ninguna coincidencia, por los factores anteriormente comentados (mucha vegetación, distinta perspectiva y pocos elementos significativos)

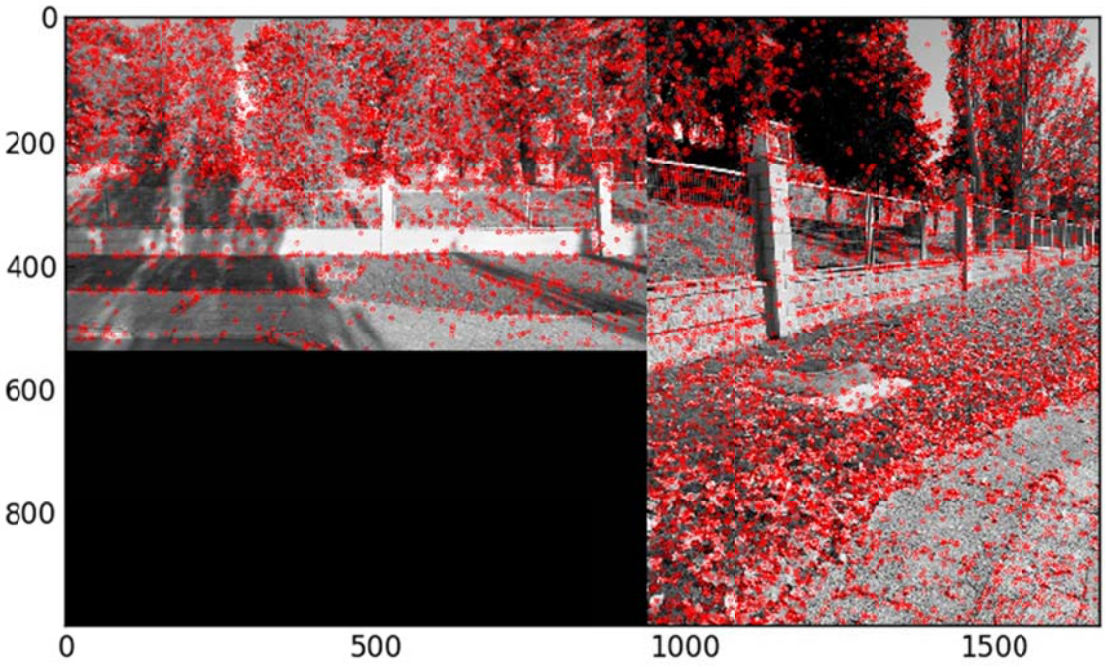
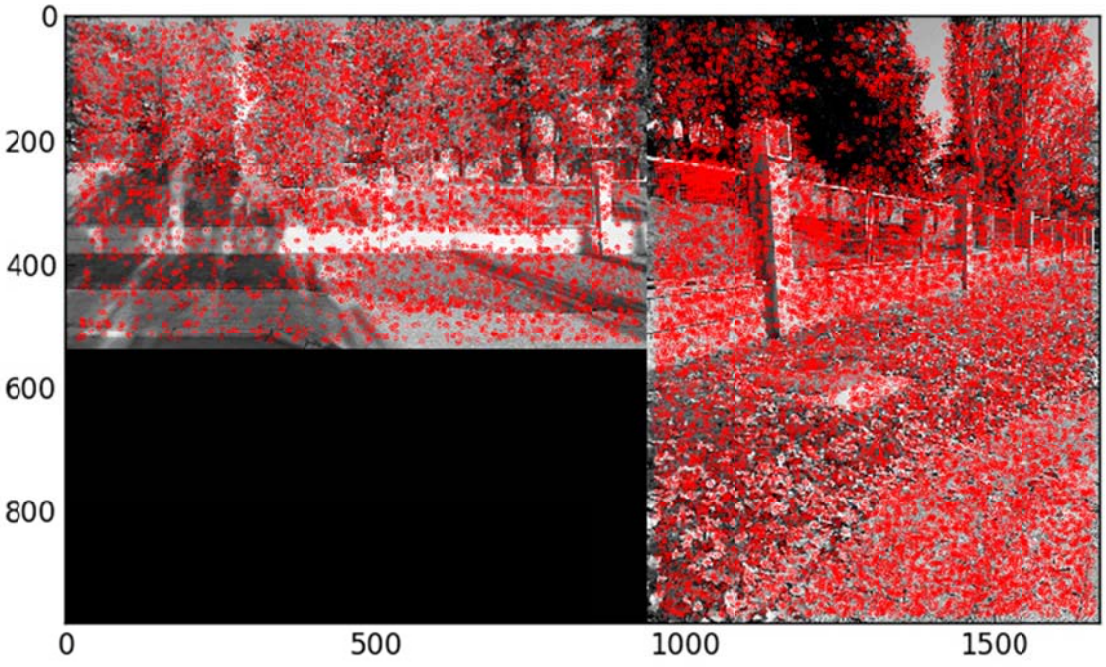
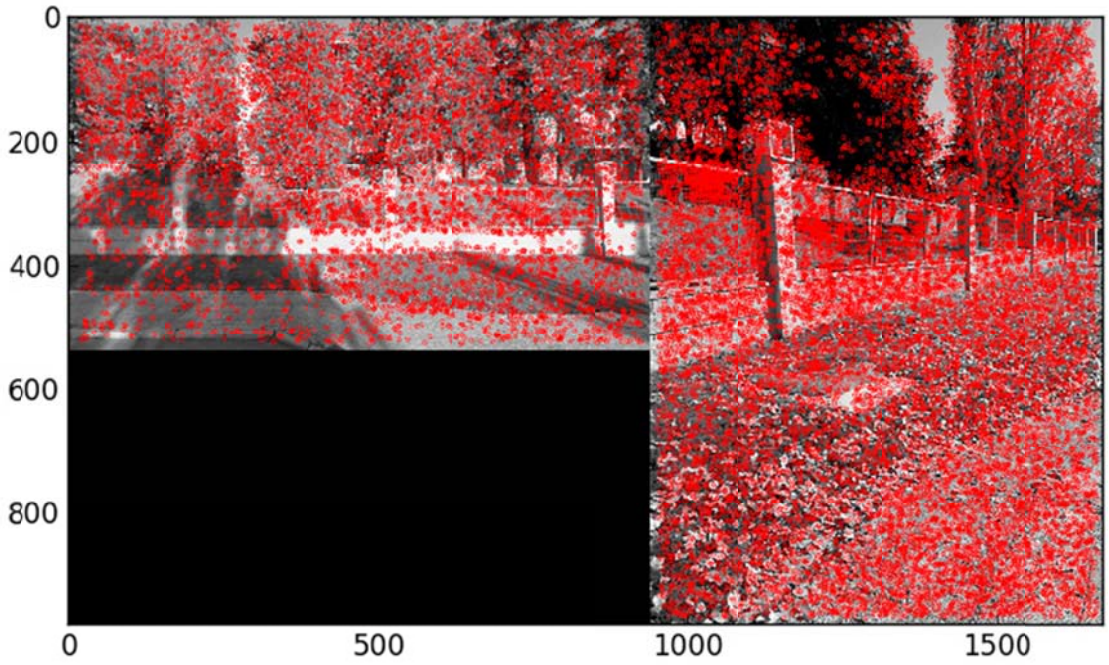


Fig. 5.15. Resultados de prueba avenida Complutense con SIFT



5.16. Resultados de prueba de avenida Complutense con SURF

En la Figura 5.16 se muestran los resultados tras aplicar SURF. El resultado es el mismo que con SIFT (no se obtienen coincidencias) aunque el tiempo de ejecución es menor (23 frente a 32).



5.2.6. Calle Aurora

En la Figura 5.17 se muestra la última prueba, en la que se presenta una superficie uniforme, en este caso un muro de ladrillo.

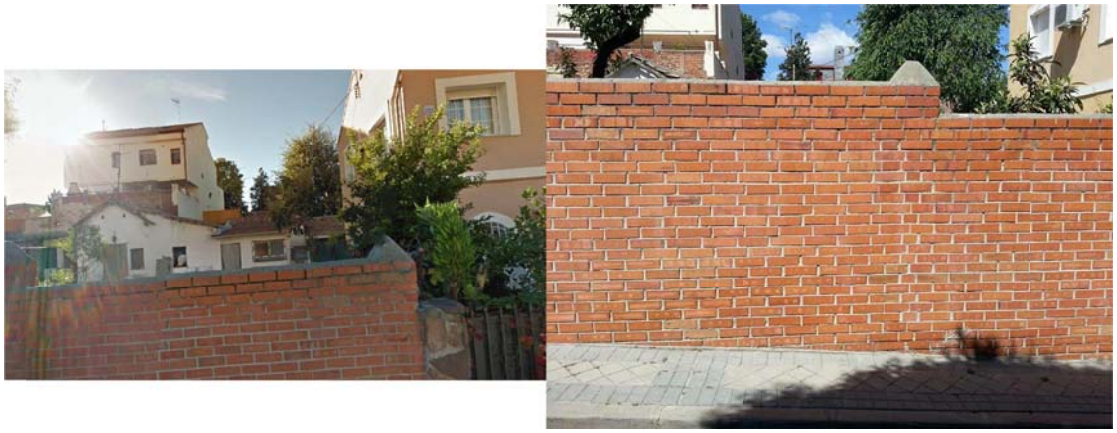


Fig. 5.17 Foto usada en prueba calle Aurora.

Como se puede comprobar, la imagen descargada de Google Street View solo ofrece la parte superior del muro, ya que es una calle estrecha por lo que las fachadas están muy cerca, ofreciendo un ángulo de visión menor por cada foto descargada. De esta manera que solo coinciden partes de la imagen, como se puede ver en la Figura 5.18:

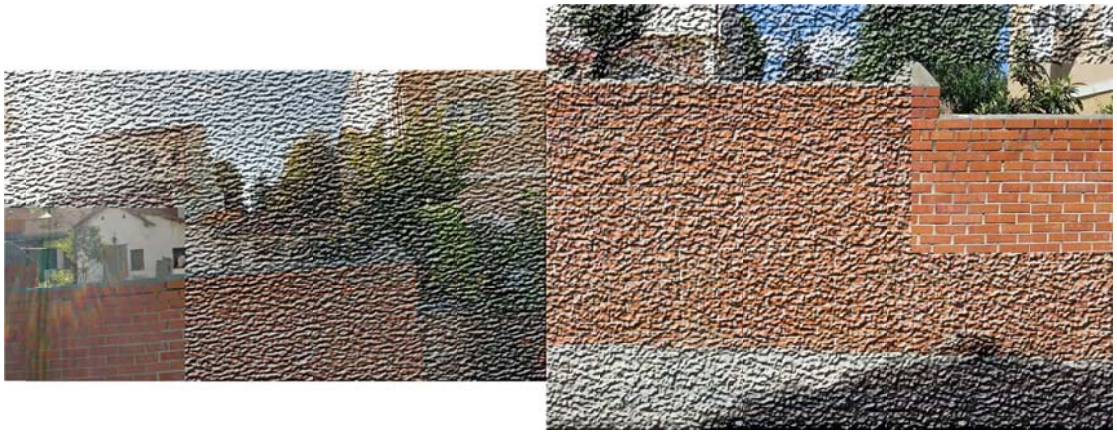


Fig.5.18. Las partes que coinciden en la foto de la calle Aurora.

En la Figura 5.19 se muestran los resultados tras aplicar SIFT, obteniendo 5 coincidencias, aunque el resultado no es muy fiable, ya que al ser una superficie uniforme, podría haber muchas posibles coincidencias.

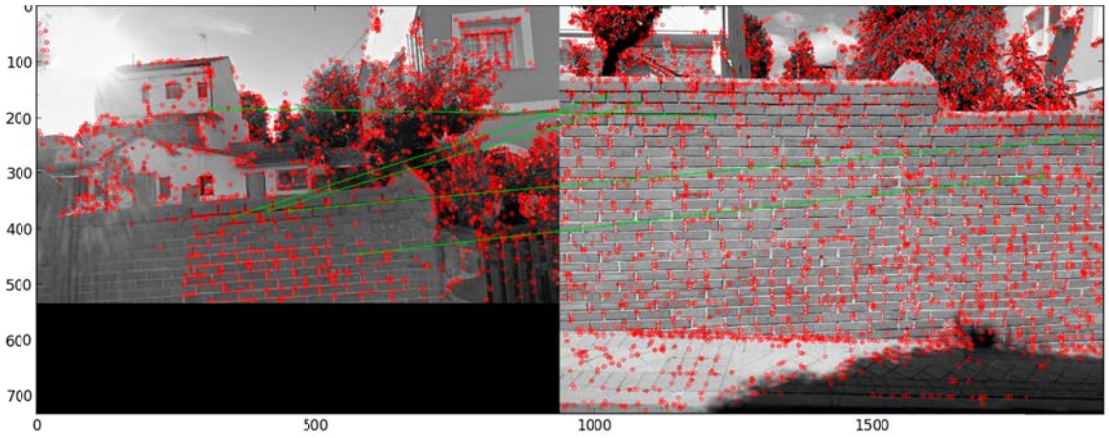


Fig. 5.19. Resultados de prueba calle Aurora con SIFT

En la Figura 5.20 se muestran los resultados con SURF, siendo estos aún peores que con SIFT, ya que no se obtiene ninguna coincidencia y en este caso el tiempo de ejecución es mayor.

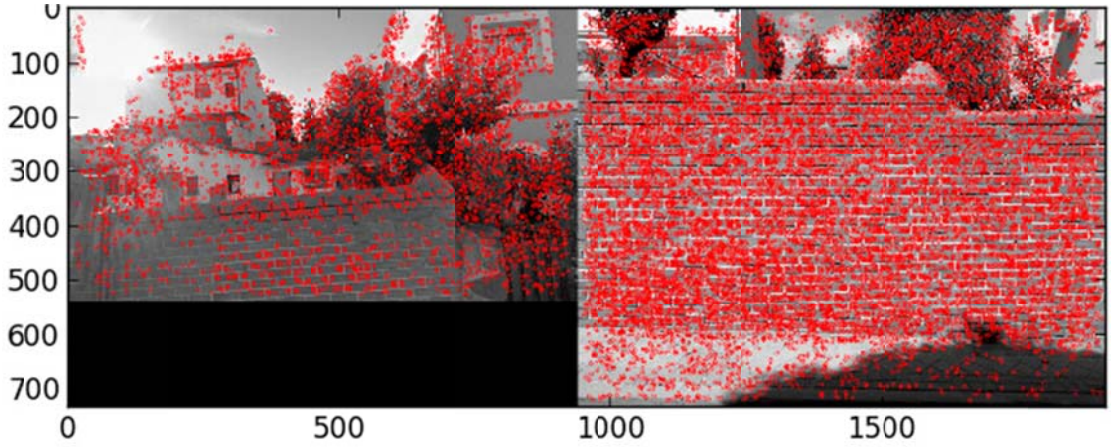


Fig. 5.20. Resultados de prueba calle Aurora con SURF

Una vez analizadas las pruebas individuales, se llega a las siguientes conclusiones:

- El algoritmo de comparación se comporta bien con escenas urbanas donde abundan los elementos significativos (rótulos, letreros, señales, ...).
- El algoritmo de comparación se comporta mal para escenas urbanas con mucha vegetación o escasez de elementos significativos.
- La perspectiva y el nivel de zoom con el que están tomadas las fotografías influye en la comparación
- La diferencia entre las fechas en las que se han tomado la imagen del usuario y la de Google Street View, puede ser determinante, al cambiar la escena urbana (cambios de tiendas, obras, ...)
- SIFT es más consistente que SURF (ofrece menos falsos positivos)
- SURF es más rápido que SIFT

Portanto, se decidió usar SIFT para los experimentos de las pruebas conjuntas.

5.3. Pruebas conjuntas

Una vez realizadas las pruebas individuales y decidir usar SIFT como el descriptor a utilizar, los siguientes experimentos son pruebas conjuntas, es

decir, dada una imagen de la cual queremos conocer su posición geográfica (latitud, longitud), se comparan los descriptores de dicha imagen con los descriptores de cada una de las fotos de la base de datos.

De esta manera, se procedió a aplicar el algoritmo creado, tomando como imágenes a localizar, imágenes utilizadas en la fase anterior de pruebas, desechando aquellas que tuvieron un mal resultado, puesto que ya se sabe que el resultado va a ser similar. En la Tabla 5.2 se resumen las pruebas realizadas, indicando en cada una de ellas, el número de imágenes de la base de datos utilizada, el tiempo aproximado de ejecución del algoritmo y la posición en la que aparece la imagen de la base de datos que corresponde a la consultada, tras ordenar las comparaciones realizadas de mayor a menor número de coincidencias.

Número	Calle	Imágenes en BD	Tiempo de ejecución	Posición Ranking
1	Calle Mayor	15	1 min	1ª
2	Isla de Oza	74	7 min	1ª
3	Isla de Oza	74	9 min	3ª
4	Isla de Oza	74	14 min	2ª
5	Isla de Oza	74	8 min	1ª
6	Isla de Oza	74	10 min	1ª
7	Calle Aurora	38	4 min	1ª
8	Calle Aurora	38	5 min	3ª

Tabla. 5.2. Resultados de las pruebas conjuntas

5.3.1. Calle Mayor II

En la Figura 5.21 se muestra la imagen que se desea geolocalizar en esta prueba.

Fig. 5.21. Foto a geolocalizar en la aplicación.

En este caso, el tiempo de ejecución del programa fue muy reducido, gracias a que la imagen a consultar (imagen superior) es bastante pequeña (400x286 px)

Los resultados obtenidos fueron muy positivos, obteniendo 14 coincidencias en la primera imagen (la correcta), y tan solo una en la segunda y en la tercera (falsos positivos). En la Figura 5.22 se muestran estas tres imágenes.

Fig. 5.22. Mejores coincidencias obtenidas ordenadas de izquierda a derecha

5.3.2. Isla de Oza IV

En la Figura 5.23 se muestra la imagen que se desea geolocalizar en esta prueba.

Fig. 5.23. Foto a geolocalizar en la aplicación de Isla de Oza IV.

En este caso, comparamos la imagen del taller con 74 fotografías de la base de datos de imágenes descargadas de Google Street View. Al tener que realizar 74 comparaciones el tiempo de ejecución se eleva hasta alrededor de los 7 minutos.

Los resultados son muy satisfactorios, consiguiendo 98 coincidencias en la imagen correspondiente, y 10 en la segunda y en la tercera. La segunda imagen aparece en el ranking ya que es un local adyacente al taller y en la imagen a consultar aparece parte de dicho local. Sin embargo, la tercera imagen tiene el mismo número de coincidencias (10) y son todos falsos positivos. Así, aunque en este caso no afecta, los falsos positivos podrían llegar a dejar fuera a la imagen que realmente corresponde. En la Figura 5.24 se muestran las tres imágenes de la base de datos con más coincidencias.

Fig. 5.24. Mejores coincidencias obtenidas ordenadas de izquierda a derecha

5.3.3. Isla de Oza V

En la Figura 5.25 se muestra la imagen que se desea geolocalizar en esta prueba.

Fig. 5.25. Imagen a geolocalizar en la aplicaciónde Isla Oza V.

En este caso, comparamos una imagen cuyos resultados en las pruebas individuales no fueron muy brillantes. El tiempo de ejecución fue de 9 minutos.

A pesar de que en la imagen correspondiente a la consultada se obtiene un número muy bajo de coincidencias (sólo 5), el bajo número de falsos positivos en las otras imágenes (menos de 5) hace que, por lo menos, la imagen correspondiente aparezca en los resultados (la tercera).

Como primera imagen aparece la misma que apareció en la 3ª posición del ranking de la prueba anterior. Es una imagen en la que se observa gran cantidad de vegetación, lo que indica que este tipo de imágenes inducen a error.

En la Figura 5.26 se muestran las tres imágenes de la base de datos con más coincidencias.

Fig 5.26. Mejores coincidencias obtenidas ordenadas de izquierda a derecha

5.3.4. Isla de Oza V

En la Figura 5.27 se muestra la imagen que se desea geolocalizar en esta prueba.

Fig 5.27. Foto a geolocalizar en la aplicación de Isla Oza V.

En esta prueba se trata de localizar a una clínica veterinaria. En este caso la imagen aparece en el segundo lugar del ranking con 24 coincidencias, por detrás de la primera, con 34, todos falsos positivos. De esta manera vemos como en esta prueba los falsos positivos hacen que la imagen "buena", no aparezca en primer lugar. En la Figura 5.28 se muestran las tres imágenes de la base de datos con más coincidencias.

Fig. 5.28. Mejores coincidencias obtenidas ordenadas de izquierda a derecha

5.3.5. Isla de Oza VI

En la Figura 5.29 se muestra la imagen que se desea geolocalizar en esta prueba.

Fig. 5.29. Foto a geolocalizar en la aplicación de Isla de Oza VI.

En esta prueba se compara otra imagen de un comercio (una peluquería). Los resultados son muy buenos obteniendo muchas más coincidencias en la imagen "buena" (91) que en la segunda y la tercera (14 y 11). Dichas imágenes se muestran en la Figura 5.30. Los buenos resultados se explican por el rótulo (muy descriptivo) y por la cercanía del comercio a la calle, lo que hizo que la imagen de Google Street View estuviera muy cerca del comercio, obteniendo un gran nivel de detalle.

Fig. 5.30. Mejores coincidencias obtenidas ordenadas de izquierda a derecha

5.3.6. Isla de Oza VII

En la Figura 5.31 se muestra la imagen que se desea geolocalizar en esta prueba.

Fig. 5.31. Foto a geolocalizar en la aplicación de Isla de Oza VII.

En esta prueba comparamos una imagen de la que se obtuvieron muy buenos resultados en las pruebas individuales. El resultado con comparación con otras imágenes fue, también, muy bueno obteniendo 91 coincidencias en el primer resultado y 14 y 11 en el segundo y tercer resultado respectivamente. En la Figura 5.32 se muestran estas tres imágenes.

Fig. 5.32 Mejores coincidencias obtenidas ordenadas de izquierda a derecha

Es de destacar, la presencia otra vez de la imagen del tercer resultado (mucha vegetación) que ya apareció en los resultados de dos pruebas anteriores.

5.3.7. Calle Aurora II

En la Figura 5.33 se muestra la imagen que se desea geolocalizar en esta prueba.

Fig. 5.33. Foto a geolocalizar en la aplicación de calle Aurora II.

En esta prueba se cambia de calle. Así se comparó la imagen con 38 imágenes de la misma calle. La imagen aparece en el primer resultado, aunque no con muchas coincidencias 19, frente a 11 y 8 de la segunda y la tercera. Esto es debido a que la escena no es muy descriptiva, aunque el buen nivel de zoom de la imagen de Google Street View hace que se obtengan buenos resultados. Las 3 imágenes con más coincidencias se muestran en la Figura 5.34.

Fig. 5.34. Mejores coincidencias obtenidas ordenadas de izquierda a derecha

5.3.8. Calle Aurora III

En la Figura 5.35 se muestra la imagen que se desea geolocalizar en esta prueba.

Fig. 5.35. Imagen a geolocalizar en la aplicación de calle Aurora III.

La última prueba es muy similar a la anterior. Sin embargo, en este caso los resultados no son tan positivos, relevando a la imagen correspondiente a la tercera posición. Las tres imágenes con más coincidencias de la base de datos se muestran en la Figura 5.36. De esta manera se puede ver como los falsos positivos en otras imágenes que no tienen nada que ver (21 y 9) hace que la imagen "buena" salga en un tercer puesto con 8 coincidencias.

Fig. 5.36. Mejores coincidencias obtenidas ordenadas de izquierda a derecha

6. CONCLUSIONES

Después de analizar el trabajo realizado se pueden extraer las siguientes conclusiones:

1. Las características y descriptores de imágenes son fundamentales en la actualidad teniendo gran cantidad de aplicaciones que van desde la monitorización de objetos en movimiento hasta la navegación de robots de forma autónoma. En este trabajo se presenta otra aplicación relevante: análisis forense de imágenes.

2. De los tipos de detectores de características los más completos y útiles para el objetivo del trabajo son los detectores de regiones, ya que proporcionan información que no pueden proporcionar los detectores de bordes y esquinas. Además este tipo de detectores son invariantes a cambios de escala, algo que no ocurre en otros detectores y que es fundamental para obtener buenos resultados.

3. Google Street View proporciona a investigadores una base de datos de imágenes de grandes dimensiones que ofrece distintas ventajas sobre otras bases de datos de imágenes tradicionales como Flickr. Una de las principales ventajas es la homogeneidad de la base de datos, dando la posibilidad de obtener imágenes de cualquier lugar, sea más o menos turístico (algo que en Flickr no ocurre). Esto hace que Google Street View sea la base de nuevos trabajos como obtener los elementos significativos de una ciudad o geolocalizar una imagen.

4. Tras la realización de 18 pruebas individuales en las que se comparaban dos imágenes entre si, aplicando SIFT y SURF, se concluye que en la mayoría de las ocasiones SURF es más rápido que SIFT (tiempo de ejecución menor). Sin embargo, SIFT obtiene resultados algo más consistentes (menos falsos positivos).

5. Tras realizar las pruebas individuales (comparar dos imágenes) y las conjuntas (comparar una imagen con cada una de las imágenes de una base de datos, se concluye que el algoritmo de comparación obtiene buenos resultados para escenas urbanas en las que abundan elementos significativos o distintivos. Estos elementos pueden ser rótulos, letreros, señales. Al contrario, en escenas donde escasean este tipo de elementos, o las superficies son uniformes, el algoritmo suele obtener peores resultados.

   Cabe destacar el caso de las escenas con mucha vegetación. Los detectores, tanto SIFT como SURF, detectan en ellas muchos puntos clave. En estos casos el algoritmo de comparación se comporta mal, arrojando, en numerosas ocasiones gran cantidad de falsos positivos y ralentizando el tiempo de ejecución.

   Por otra parte, tras analizar las pruebas, se puede concluir que la perspectiva y el nivel de zoom con el que están tomadas las fotografías que se quieren geolocalizar, influye en los resultados. De esta manera se obtienen mejores resultados para fotografías que se han tomado cerca y que tienen una perpectiva bastante similar a la de Google Street View (en paralelo).

   Finalmente, es de destacar que la diferencia entre las fechas en las que se han tomado la imagen del usuario y la de Google Street View puede ser determinante si cambia la escena urbana en ese periodo de tiempo (cambios de tiendas, obras, ...)

REFERENCIAS

[1] Dorkó, G. and Schmid, C., "Selection of Scale-Invariant Parts for Object Class Recognition", *Proceedings Ninth IEEE International Conference on Computer Vision,* pp. 634-639, 2003.

[2] Leibe, B. and Schiele, B., "Scale-Invariant Object Categorization Using a Scale-Adaptive Mean-Shift Search". *Pattern Recognition,* pp. 145-153, 2004.

[3] Cedillo, J. A. Á., Lozada, J. C. H. and Carbajal, M. O., "Comparativo de las Técnicas de Reconstrucción 3D para Imágenes Médicas", *Biociencias y Nanociencias*, SINNCO 2010.

[4] Liu, J., Luo, J., & Shah, M., "Recognizing Realistic Actions From Videos 'in the wild'". proceedings of the *IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1996-2003, 2009.

[5] Nowak, E., Jurie, F. and Triggs, B., "Sampling Strategies for Bag-of-Features Image Classification" *Computer Vision–ECCV* pp. 490-503, 2006.

[6] Mikolajczyk, K.; Leibe, B. & Schiele, B., "Local Features for Object Class Recognition", *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV), 2, pp. 1792-1799, 2005.*

[7] OpenCV -Python Tutorials, http://opencv-python-tutroals.readthedocs.org/

[8] https://courses.cs.washington.edu/courses/cse455/09wi/Lects/lect6.pdf

[9] Wah, Benjamin W. Wiley *Encyclopedia of Computer Science and Engineering*. Wiley-Interscience, 2007.

[10] J. M. Park and Y. Lu. "Edge Detection in Grayscale, Color and Range Images", B. W. Wah (editor) Encyclopedia of Computer Science and Engineering, 2008.

[11] L. Kitchen and A. Rosenfeld, "Gray-Level Corner Detection", *Pattern Recognition Letters* **1** (2). pp. 95–102, 1982.

[12] T. Lindeberg and J.-O. Eklundh, Scale Detection And Region Extraction From A Scale-Space Primal Sketch", *Proceedings of the 3rd International Conference on Computer Vision*, (Osaka, Japan), pp. 416-426, Dec. 1990.

[13] Mikolajczyk, K. and Schmid, C.,. "Scale & affine invariant interest point detectors". *International journal of computer vision*, *60*(1), pp. 63-86, 2004.

[14] Canny, J.: "A Computational Approach to Edge Detection" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 679- 698, 1986.

[15] Harris, C. and Stephens, M, "A Combined Corner and Edge Detector" *Proceedings of the Conference on Alvey vision,* pp. 50, 1988.

[16] Lowe, D. G., "Distinctive Image Features from Scale-Invariant Keypoints", *International Journal of Computer Vision*, *60*(2), pp. 91-110, 2004.

[17] Bay, H., Tuytelaars, T. and Van Gool, L., "Surf: Speeded Up Robust Features", *Computer Vision–ECCV 2006* pp. 404-417, 2006.

[18] S. Mallat, "A Wavelet Tour of Signal Processing, Academic Press, San Diego, CA 92101-4495, USA, 1998.

[19] Crow, Franklin, "Summed-Area Tables for Texture Mapping", *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*. pp. 207–212, 1984.

[20] Shlens, J., "A Tutorial on Principal Component Analysis Derivation, Discussion and Singular Value Decomposition", 25 March 2003, Version 1.

[21] Zhu, Q., Yeh, M. C., Cheng, K. T. and Avidan, S., "Fast Human Detection Using a Cascade of Histograms of Oriented Gradients", *Computer Vision and Pattern Recognition,* 2, pp. 1491-1498, 2006.

[22] API de Imágenes de Google Maps, https://developers.google.com/maps/ documentation/ streetview/?hl=es, 2014

[23] Google Street View Static API, http://jamiethompson.co.uk/web/2010/05/15/google-streetview-static-api/

[24] Google Street View: Tiles, http://geo.jamiethompson.co.uk/streetview_tiles.php?postcode=SW1W+9TQ&zoom=4

[25] Newton's Cannon, http://www.newtonscannon.com/2014/01/26/capturing-spherical-scenes-from-google-streetview/

[26] Doersch, C., Singh, S., Gupta, A., Sivic, J. and Efros, A. A., "What Makes Paris Look Like Paris?", *ACM Transactions Graph.*, *31* (4), 2012.

[27] Sattler, T., Leibe, B., and Kobbelt, L. "Fast Image-Based Localization Using Direct 2D-to-3D Matching", *Proceedings of the IEEE International Conference on Computer Vision (ICCV),* pp. 667-674, *2011.*

[28] Salmen, J.; Houben, S.; Schlipsing, M., "Google Street View Images Support the Development of Vision-Based Driver Assistance Systems", *Intelligent Vehicles Symposium (IV),* 3 (7), pp. 891-895, 2012.

[29] Zamir, A. R., and Shah, M., "Accurate Image Localization Based On Google Maps Street View", *Computer Vision-ECCV 2010* pp. 255-268, 2010.

[30] Picard, D., Cord, M. and Valle, E., "Study of SIFT Descriptors For Image Matching Based Localization in Urban Street View Context", Stilla U, Rottensteiner F, Paparoditis N (Eds) CMRT09. IAPRS, XXXVIII, Part 3/W4 Paris, France, 3-4 September, 2009.

[31] Valle, E., "Local-Descriptor Matching for Image Identification Systems". PhD thesis, Univ. Cergy-Pontoise, ETIS, UMR, CNRS 8051, 2008.

[32] Fischler and Bolles, RANSAC SRI International ,1981.

[33] Building Streetview Datasets for Place Recognition and City Reconstruction, Gronat, 2011.

[34] Github, https://github.com/eligrey/FileSaver.js/

[35] Interprocess Communication and Networking, https://docs.python.org/2/library/subprocess. html

[36] Python Object Serialization, https://docs.python.org/2/library/pickle.html

[37] Fast Library for Approximate Nearest Neighbors http://www.cs.ubc.ca/research/flann/

[38] Friedman, J. H., Bentley, J. L. and Finkel, R. A., "An Algorithm for Finding Best Matches in Logarithmic Expected Time", *ACM Transactions on Mathematical Software (TOMS)*, *3*(3), pp. 209-226, 1977.

ANEXO I: DESCRIPCIÓN DE LA HERRAMIENTA

En este anexo se procede a explicar la herramienta diseñada en este trabajo.

II.1 Pantalla Principal

Al ejecutar la herramienta se abre la pantalla principal de la herramienta, como se muestra en la Figura A.1. En la parte izquierda cuenta con un explorador del sistema de ficheros, con el que se puede seleccionar, de manera sencilla, la imagen de la cual queremos saber su posición GPS. En la parte derecha, se encuentra un visualizador que muestra la imagen seleccionada en el explorador.

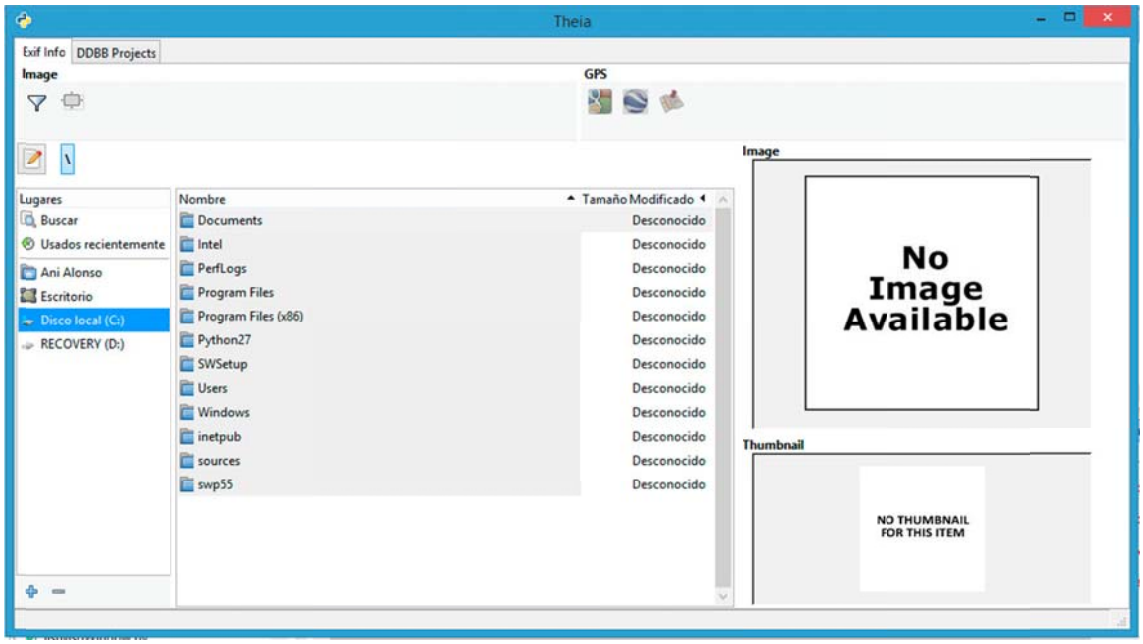


Fig. A.1. Pantalla principal de la herramienta

En la Figura A.2 se muestra el comportamiento al seleccionar una imagen, mostrándose los metadatos de la imagen, si es que tiene y habilitándose el botón de la derecha de la parte superior de la Figura.

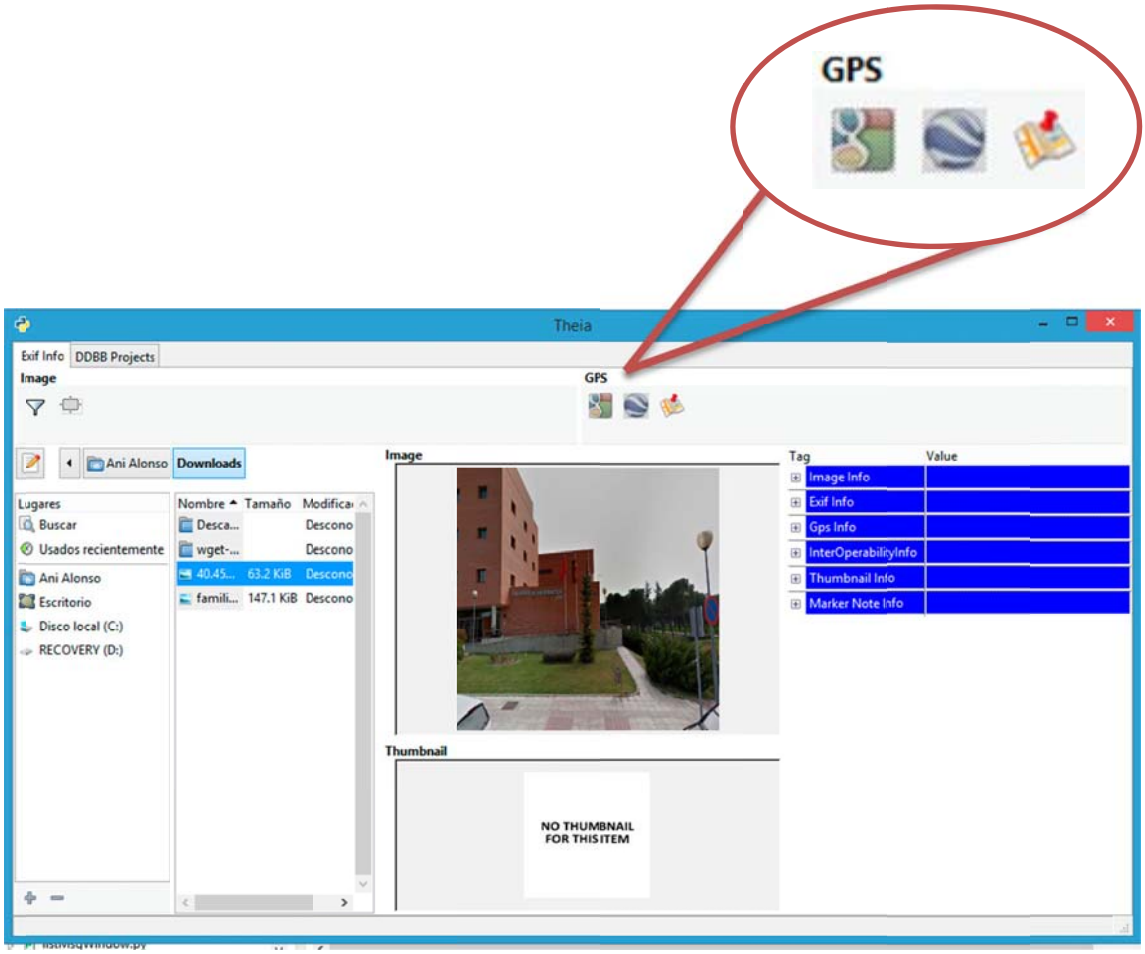


Fig.A2. Pantalla principal al seleccionar una imagen del sistema.

II.2 Pantalla de Geolocalización de Imágenes

Al hacer clic sobre el botón de la derecha de la Figura A.3, se abre una nueva ventana. Como se puede ver en la Figura A.4, en la ventana se muestra la ruta de la imagen seleccionada, la propia imagen y tres botones que representan los tres pasos necesarios para geolocalizar la imagen.

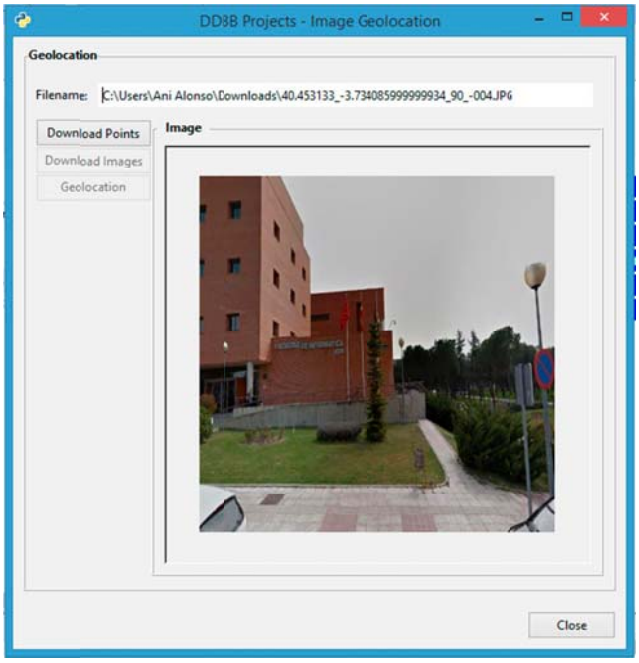


Fig. A.4. Pantalla de geolocalización de imágenes

Al hacer clic en el primer botón ("Download points"), se abre una nueva pestaña en el navegador con la página web explicada en el apartado 4.3.1.1, para descargar los id de las imágenes correspondientes a los puntos del mapa que el usuario elige.

Como se puede ver en la Figura A.5, la página web consta de un mapa en la parte izquierda donde el usuario puede elegir los puntos a descargar. Sobre ese mapa, se sitúa una caja de texto, donde el usuario puede buscar un sitio (calle, pueblo, ciudad, ...) para posicionarse de forma sencilla.

Fig. A.5. Página web de obtención de los puntos de las imágenes a descargar.

Este comportamiento se puede ver en la Figura A.6. En la parte derecha se sitúa otra caja de texto en la que el usuario debe introducir un nombre cualquiera que identifique a esa descarga, unas instrucciones y un botón para obtener todos los puntos.

Fig. A.6. Funcionamiento de la caja de texto para posicionarse en el lugar deseado

En la Figura A.7 se muestra el comportamiento tras presionar el botón de "Obtener puntos para descargar". El progreso de la tarea se refleja con una barra de progreso azul entre el título de la página y el mapa.

Fig. A.7. Progreso de la obtención de los puntos

Una vez terminada la tarea se habilitan dos botones ("Descargar mapping.txt" y "download.txt"), como se aprecia en la Figura A.8 con los que el

usuario puede guardar los ficheros generados en disco.

Fig. A.8. Botones de descarga de los archivos "mapping.txt" y "download.txt" habilitados

Una vez realizado el paso 1 de descarga de los puntos, se habilita el botón "Download Images", encargado de descargar las imágenes con Matlab, extraer los descriptores y almacenarlos en disco. De esta manera, tal y como se muestra en la Figura A.9 se abre la consola de matlab, mostrando las trazas de la ejecución del programa que descarga las imágenes.

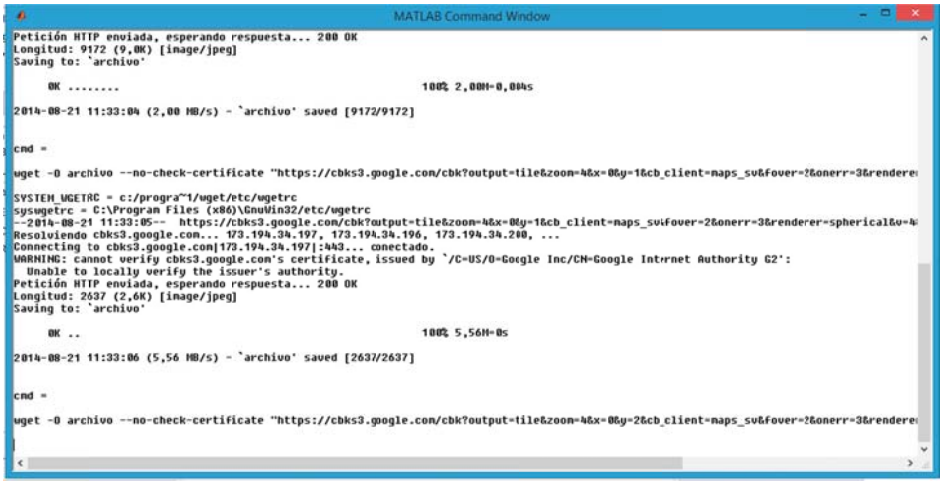


Fig. A.9. Vista de la consola de Matlab descargando las imágenes.

Finalmente se habilita el botón "Geolocation", encargado de comparar la imagen seleccionada con todas las de la base de datos haciendo uso de los

descriptores. En la Figura A.10. se muestra la salida del programa en Eclipse.

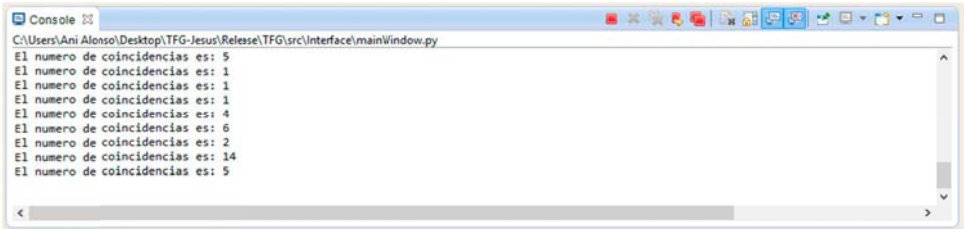


Fig. A.10. Vista de la salida del programa en la consola de Eclipse

II.3 Pantalla de Resultados

En la Figura A.11 se muestra la nueva ventana que aparece una vez terminada la comparación. En ella aparecen los 10 mejores resultados obtenidos, es decir, las imágenes de Google Street View con más coincidencias con la imagen seleccionada.

En la parte izquierda se tiene una lista con los resultados. Por cada entrada se muestra el número del ranking, la latitud y longitud de la imagen y el número de coincidencias.

En la parte derecha aparecen tres imágenes. La superior es la imagen original, la central es la de Google Street View, y la inferior es la comparativa entre ambas imágenes. Estas imágenes cambian en función de la entrada de la lista que esté seleccionada.

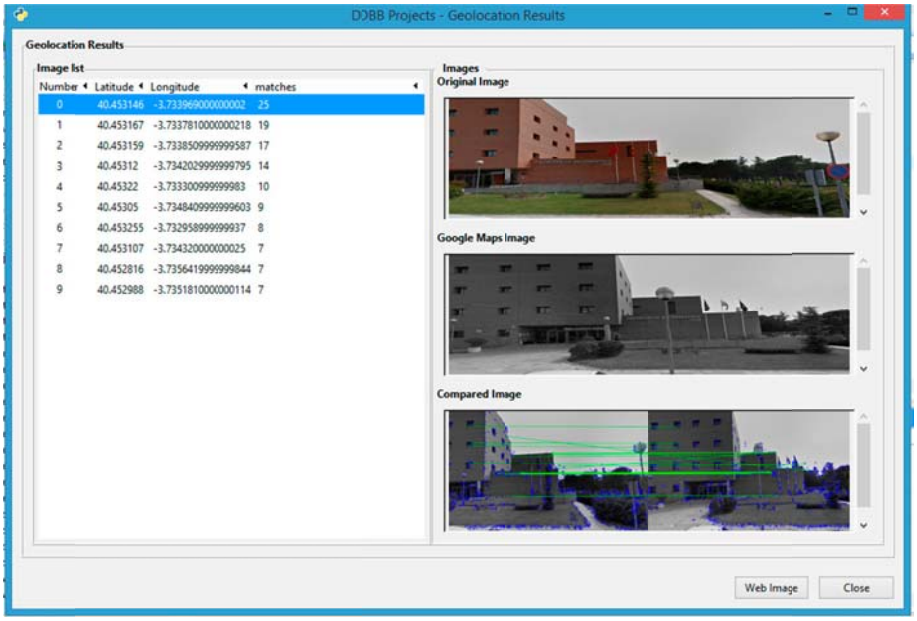


Fig. A.11. Pantalla de resultados

Por último en la parte inferior aparecen dos botones: Uno para cerrar la ventana y otro para mostrar la localización de la imagen en Google Maps. La Figura A.12 muestra este caso.

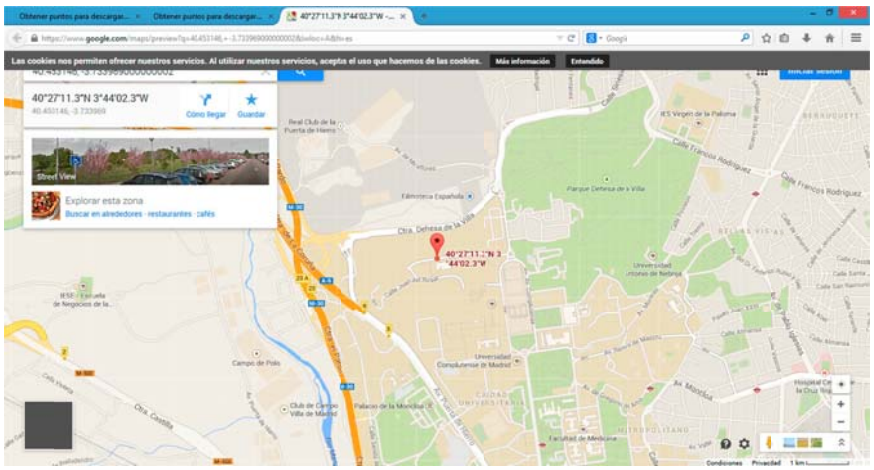


Fig. A.12. Vista de la posición en Google Maps de unas de las imágenes con más resultados